Stéphane Grare

Les fichiers Batch

Concevez rapidement des fichiers batch en utilisant les commandes MS-Dos...



Stéphane Grare

Un fichier Batch est un fichier qui regroupe une suite logique de commandes MS-Dos, et possédant l'extension ".bat" et est directement exécutable par le système. Les commandes sont enregistrées ligne par ligne et seront exécutées séquentiellement. La programmation Batch nécessite une connaissance minimum de l'environnement Dos. En fait, un fichier Batch contient simplement une suite de commandes que vous pourriez taper sous l'invité (prompt) du Dos, chaque nouvelle ligne du fichier correspondant à une nouvelle commande. Néanmoins, certaines commandes ne sont utilisables que dans les fichiers Batch du fait de leur inutilité dans l'environnement de commande Dos. C'est ce qui permet d'automatiser certaines tâches. Leur utilité est, par exemple, quand il faut répéter toujours la même série de commandes. À titre d'exemple, nous pourrions évoquer le changement de répertoire et peut-être aussi la commande Format qu'on fait souvent suivre de la commande CHKDSK pour vérifier si la disquette a bien été formatée.

INTRODUCTION

L'intérêt des Batchs est donc d'automatiser des tâches répétitives effectuées sous DOS. Les fichiers Batch peuvent également utiliser toutes les commandes DOS, ce qui rend disponible pour le programmeur un grand nombre de fonctions. Enfin, leur taille est relativement légère par rapport à d'autres programmes, ce qui facilite leurs "transferts" sur différents disques et supports de stockage. Cependant…

· Le langage Batch n'est pas compilé, il est interprété par COMMAND.COM ce qui rend plus lent l'exécution de programmes Batch par rapport à des applications écrites directement en langage machine,

· Les fichiers Batch sont directement éditables, donc votre code n'est pas "protégé" à la copie par d'autres programmeurs,

· Enfin, et surtout, des opérations élémentaires telles que le traitement de chaînes de caractères, d'opérations mathématiques... n'existent pas sous DOS, ce qui implique l'usage de programmes externes (s'ils existent, selon les cas).

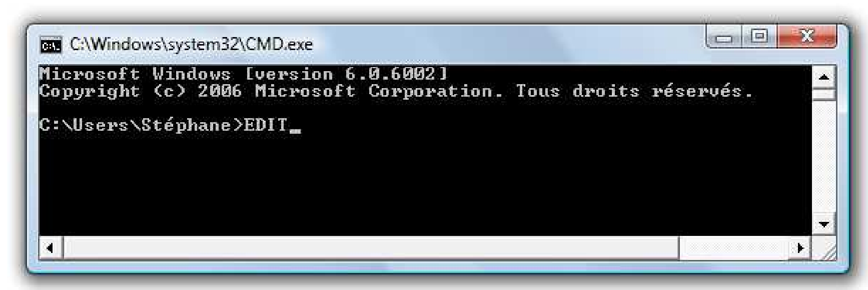
TABLE DES MATIERES

1 – <u>Création d'un fichier Batch</u> :

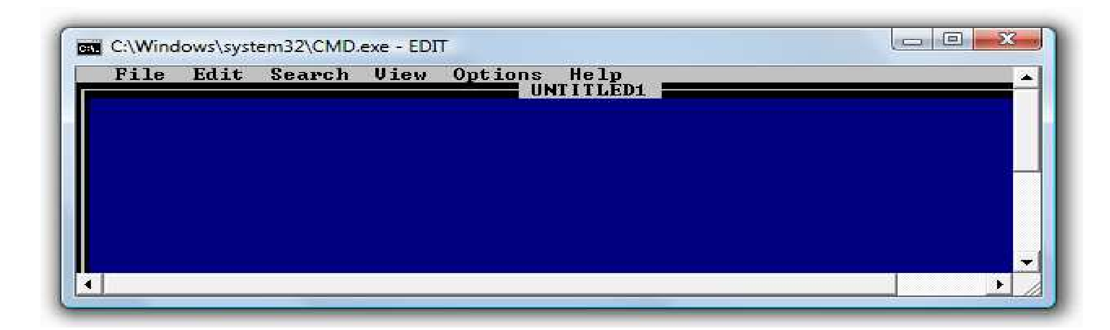
Le terme "Batch" désigne un fichier contenant une suite de commandes qui seront traitées automatiquement. Nous appelons aussi cela un "traitement par lot". Afin de créer votre premier fichier Batch, suivez cette procédure :

1) Ouvrez un éditeur de texte : Le Bloc-notes par exemple.
2) Inscrivez vos lignes de commandes.
3) Enregistrez votre fichier texte.
4) Cliquez sur le nom du fichier puis appuyez sur la touche F2. Vous serez en mode "Édition".
5) Renommez le fichier en changeant l'extension .txt en .bat. À la question : "Voulez-vous vraiment renommer le fichier", répondez par Oui. Attention de désactiver au préalable la case "Masquer les extensions de fichiers dont le type est connu" dans les options avancées de l'Explorateur Windows.

Pour créer un fichier Batch, vous pouvez également utiliser directement l'interpréteur de commandes : pour cela, utilisez la commande **EDIT** :



Appeler alors la commande **EDIT** pour l'éditeur de texte :



2 – <u>Notions de base sur les fichiers Batch</u> :

Malheureusement, la majorité des sources intéressantes qui pourront vous aider à approfondir les techniques de la programmation Batch sont souvent en anglais... "Xset" est un outil indispensable pour appendre à écrire des fichiers Batch sous toutes les versions de Windows. De plus, Marc Stern propose de nombreux didacticiels à partir de son site Internet : http://xset.tripod.com/.

Jerold Schulman qui m'a autorisé à citer quelques-uns de ces scripts anime un site Internet qui propose d'innombrables pages sur la programmation Batch : http://www.jsiinc.com/. C'est, par ailleurs, un des meilleurs sites traitant des systèmes NT de Windows.

D'excellents scripts sont également visibles à partir de cette adresse : www.uwasa.fi/~ts/http/http2.html#batch. Timo Salmi propose une multitude de solutions permettant de résoudre les problèmes les plus courants quand on commence à se lancer dans la programmation Batch.

Ritchie Lawrence met en lignes de nombreux modèles de scripts et propose quelques utilitaires fort bien faits qui vous permettront de réaliser des choses étonnantes en mode Ms-Dos ou en Invite de commandes. L'adresse de son site est la suivante : http://www.commandline.co.uk/index.html.

Enfin, une source inépuisable de trouvailles est proposée à partir de ce forum recouvrant une multitude de thématiques dont Ms-Dos et l'Invite de commandes : http://www.experts-exchange.com/Operating_Systems/MSDOS. La plupart des exemples de scripts expliqués dans ce chapitre s'inspirent directement ou indirectement des techniques présentées par ces différents auteurs.

<u>Quelle différence entre l'extension .cmd .btm et .bat</u> ?

Un fichier .**cmd** ou **.btm** ne sera pas reconnu en tant que tel par Windows 9X. Nous pouvons voir cela comme une sorte de

garde-fou si tel fichier de commande est parfaitement incompatible avec ces versions de Windows. Par ailleurs, le processus est légèrement différent : dans le cas d'un fichier **.bat** ou **.cmd**, chaque ligne du fichier est exécutée individuellement et le fichier fermé puis ouvert à chaque lecture d'une nouvelle commande. Dans le cas d'un fichier **.btm**, le fichier n'est ouvert qu'une fois, puis lu en mémoire et enfin fermé. C'est donc a priori le mode le plus rapide. Surtout si ce sont des commandes internes qui sont exécutées.

Se servir des parenthèses dans les fichiers de scripts :

Si vous souhaitez rediriger le résultat de différentes commandes dans un même fichier texte. Par exemple :

```
@echo off
DIR /s /b *.doc >> resultat.txt
DIR /b *.dot >> resultat.txt
REM etc.
@echo on
```

Il est dans ce cas plus simple de saisir :

```
@echo off
 (
 DIR /s /b *.doc
 DIR /s /b *.dot
 ) >> resultat.txt
@echo on
```

Cela vous évitera de spécifier plusieurs fois le même fichier de sortie. Par ailleurs, c'est une manière de créer de toutes pièces un fichier texte :

```
@echo off
(
@echo Bonjour,
@echo Tout le monde !
) > test.txt
```

Dans ce dernier cas, les commandes sont regroupées.

Afficher le code de sortie :

À chaque fois que vous saisissez une commande, cette dernière produit un code de sortie qui est principalement :

1 : La commande a renvoyé une erreur
0 : La commande n'a pas renvoyé d'erreur.

C'est la variable %Errorlevel% qui est chargée de suivre les fluctuations de la réussite ou non des commandes exécutées. La commande "Set" teste les "Errorlevel" en partant de la plus petite valeur puis en procédant par incrémentation de 1. La commande "**GOTO**" démarre de la plus grande valeur puis procède par décrémentation de 1.

Gérer les caractères accentués :

Les différences entre la norme Ansi (Windows) et OEM (Ms-Dos) font que si vous vous servez d'un éditeur de texte comme le Bloc-Notes les caractères accentués ne s'afficheront pas correctement dans les sorties écran en mode "Console". Ainsi, la phrase "Je répète : "Vous êtes un entêté" s'affichera comme cela : Je rÚpÞte : "Vous Ûtes un entÛtÚ". Il y a différentes solutions :

Vous pouvez toujours vous servir de l'éditeur de texte prévu dans toutes les versions de Windows : "**Edit**".

Il existe de nombreux programmes vous permettant d'opérer la conversion Ansi vers OEM et vice-versa. "OEMANSI.exe" est un programme écrit en français que vous pouvez télécharger à partir de cette adresse :

http://perso.wanadoo.fr/andre.araste/tele2.htm

Décompressez l'archive ZIP nommée Oemansi.zip puis double-cliquez sur ce fichier exécutable : Setup.exe afin de lancer l'installation de ce logiciel. Imaginons que notre fichier à convertir s'appelle Test.bat suivez la procédure suivante :

1) Cliquez sur Démarrer/Tous les programmes/OEMANSI/OEMANSI.
2) Cliquez sur le bouton Ouvrir Fichier.
3) Dans la liste déroulante Fichiers de type :, sélectionnez Tous les fichiers (*.*) puis votre fichier Batch.
4) Cochez le bouton radio ANSI(Win) puis la flèche dirigée vers le bas.

Il est possible d'effectuer l'opération inverse !
5) Cliquez sur le bouton Sauvegarde OEM puis lancer le processus de conversion.
6) Confirmez le remplacement ou non de l'ancienne version.

Gérer les caractères réservés :

Ces caractères ne peuvent pas être employés "tels quels" dans un script. Le signe % pour être "compris" dans un fichier de script doit être redoublé. Par ailleurs, nous avons déjà vu que pour afficher un caractère réservé il faut le faire précéder du signe ^. Dans ce cas-là, il ne sera pas assimilé à une commande, mais bien à un caractère. Afin d'afficher les signes : ! ^ & < > >> " | créez un fichier Batch contenant cette commande :

```
@echo off
ECHO ^! ^% ^^ ^& ^< ^> ^>^> ^" ^|
@echo on
```

Une manière d'afficher les caractères de redirection :

```
@echo off
<nul (SET /p z=Le^|texte^|de^|sortie) >sortie.txt
@echo on
```

Nous redirigeons le produit de la commande "**SET**" vers le fichier Sortie.txt, mais en désactivant toute sortie-écran. L'utilisation de la commande "**SET**" sera expliquée plus loin.

3 – Liste des commandes :

Dans la rubrique suivante, nous détaillerons certaines commandes spécifiques. Il s'agit ici d'un aperçu rapide des commandes existantes en batch.

ANSI.SYS	Définit les fonctions qui modifient l'affichage, contrôlent le déplacement du curseur et réaffectent les touches.
APPEND	Permet aux programmes d'ouvrir les fichiers de données qui se trouvent dans les répertoires spécifies, comme s'ils figuraient dans le répertoire en cours.
ARP	Affiche, ajoute, et supprime les informations arp des dispositifs du réseau.
ASSIGN	Permet d'assigner une nouvelle lettre à un lecteur
ASSOC	Affiche les associations de fichier.
AT	Programme une heure pour exécuter une commande.
ATMADM	Liste les connexions et adresses vu par Windows ATM call manager.
ATTRIB	Affiche et change les attributs de fichiers.
BATCH	Fichier qui exécute une série de commande.
BREAK	Active / désactive CTRL + C dispositif
CACLS	Affiche et modifie le fichier ACL.
CALL	Appelle un fichier batch à partir d'un autre fichier batch.
CD	Change de répertoire.
CHCP	Permet de changer le jeu de caractères.
CHDIR	Idem que CD.
CHKDSK	Vérifie que le disque dur en format FAT ne contient pas d'erreurs.
CHKNTFS	Vérifie que le disque dur en format NTFS ne contient pas d'erreurs.
CHOICE	La commande choice demande à l'utilisateur de saisir une des lettres proposées. La récupération du choix se fait par la commande errorlevel.

CLS	Efface l'écran.
CMD	Ouvre la fenêtre de commande DOS.
COLOR	Change les couleurs de l'arrière-plan et de la police de la fenêtre
COMMAND	Idem que CMD
COMP	Compare des fichiers.
COMPACT	Compresse et décompresse des fichiers
CONTROL	Ouvrez les icônes de panneau de commande de la fenêtre DOS.
CONVERT	Converti du format FAT vers NTFS.
COPY	Copie un ou plusieurs fichiers vers une différente destination.
CTTY	Modifie les périphériques et d'entrées standards
DATE	Affiche ou modifie la date système.
DEBUG	Utilitaire permettant des programmes en assembleur afin de modifier les paramètres Hardware. Debug utility to create assembly programs to modify hardware settings.
DEFRAG	Permet de défragmenter un disque dur.
DEL	Supprime un ou plusieurs fichiers.
DELETE	Idem que DEL.
DELTREE	Efface un ou plusieurs fichiers et/ou répertoires
DIR	Affiche la liste des fichiers et des sous-répertoires d'un répertoire.
DISABLE	Désactive les services ou drivers windows.
DISKCOMP	Compare les contenus de deux disquettes.
DISKCOPY	Copie le contenu d'une disquette sur une autre.
DOSKEY	Modifie les lignes de commande, rappelle des commandes Windows, et permet de créer des macros.
DOSSHELL	Représente les répertoires et les fichiers sous forme d'icônes dans un environnement de menus déroulants qui contiennent les principales commandes du système d'exploitation.
DRIVPARM	Redéfinit les paramètres d'un lecteur.
ECHO	Affiche des messages… l'écran ou active/désactive l'affichage des commandes.
EDIT	Affiche et édite des fichiers.
EDLIN	Idem que EDIT.
EMM386	Charge extended Memory Manager.
ENABLE	Désactive les services ou drivers windows
ENDLOCAL	Stoppe la localisation des modifications de l'environnement dans un fichier de commandes.
ERASE	Supprime un ou plusieurs fichiers.
EXIT	Quitte l'interpréteur de commandes (CMD.EXE).
EXPAND	Décomcodesse un fichier.
EXTRACT	Utilitaire de décomcodession des fichiers archives cab de windows 95.
FASTHELP	Affiche des informations sur les commandes de Windows.

FC	Compare deux fichiers ou groupes de fichiers, et affiche les différences entre eux..
FDISK	Configurer / partitionner un disque dur.
FIND	Cherche une chaîne de caractères dans un ou plusieurs fichiers.
FINDSTR	Cherche des chaînes de caractères dans un ou plusieurs fichiers.
FIXBOOT	Ecrit un nouveau secteur de boot.
FIXMBR	Ecrit un nouveau secteur de boot sur un lecteur..
FOR	Exécute une commande sur chaque fichier d'un groupe de fichiers.
FORMAT	Formate un disque pour utilisation avec Windows.
FTP	Commande pour se connecter et opérer sur un serveur FTP.
FTYPE	Affiche ou modifie les types de fichiers utilisés dans les associations d'extensions.
GOTO	Poursuit l'exécution d'un fichier de commandes à une ligne identifiée par une étiquette.
GRAFTABL	Permet à Windows d'afficher un jeu de caractères en mode graphique..
HELP	Affiche des informations sur les commandes de Windows.
IF	Effectue un traitement conditionnel dans un fichier de commandes.
IFSHLP.SYS	Gestionnaire de fichier 32-bit.
IPCONFIG	Commande réseau pour voir / configurer les paramètres réseau.
KEYB	Charger un pilote de clavier.
LABEL	Créé modifie ou supprime le nom de volume d'un disque.
LH	Charge un programme en zone de mémoire supérieure, ce qui libère de la mémoire conventionnelle pour d'autres programmes.
LISTSVC	Affiche tous les services, pilotes et types de démarrage.
LOADFIX	Charge un programme sur les codemiers 64k.
LOADHIGH	Charge un programme résident dans la mémoire supérieure.
LOCK	Verrouille l'accès à un disque par une application
LOGON	Cette commande affiche les installations de Windows et de Windows NT détectées, et demande le mot de passe de l'administrateur local pour la copie de Windows à laquelle vous souhaitez vous connecter. NB: Au bout de 3 échecs, la console s'arrête et l'ordinateur redémarre...
MAP	Affiche le nom d'un lecteur.
MD	Crée un répertoire.
MEM	Affiche la mémoire du système.
MKDIR	Crée un répertoire.
MODE	Configure un périphérique du système.
MORE	Affiche la sortie écran par écran.
MOVE	Déplace un ou plusieurs fichiers d'un répertoire à un autre.
MSAV	Microsoft anti-virus.
MSD	Utilitaire de Diagnostics.
MSCDEX	Utilitaire permettant d'utiliser, de charger et accéder à un CD-ROM.

NBTSTAT	Mise à jour du cache du fichier Lmhosts
NET	Mettre à jour, réparer, ou voir le réseau et les paramètres réseau.
NETSH	Configure les informations d'un réseau dynamique et statique.
NETSTAT	Afficher les statistiques de protocole et l'état actuel des connexions NetBIOS sur TCP/IP
NLSFUNC	Charge un jeu de caractère spécifique.
NSLOOKUP	Permet de tester un serveur DNS.
PATH	Affiche ou définit le chemin de recherche des fichiers exécutables.
PATHPING	Outil de trace qui combine les caractéristiques du ping et de tracert avec des informations additionnelles qu'aucune de ces 2 commandes ne fournit.
PAUSE	Interrompt l'exécution d'un fichier de commandes et affiche un message
PING	Affiche les informations de connexion à une autre machine.
POPD	Restaure la valeur précédente du répertoire courant enregistré par PUSHD.
POWER	Conservez la puissance d'un ordinateur portable.
PRINT	Imprime un fichier texte.
PROMPT	Modifie l'invite de commande de Windows.
PUSHD	Enregistre le répertoire courant puis le modifie.
QBASIC	Ouvre QBasic.
RD	Supprime un répertoire vide.
REN	Renomme un ou plusieurs fichiers.
RENAME	Renomme un ou plusieurs fichiers.
RMDIR	Supprime un répertoire vide.
ROUTE	Manipule les tables de routage du réseau
RUNAS	Autorise un utilisateur à exécuter un programme un autre ordinateur.
SCANDISK	Lance l'utilitaire scandisk.
SCANREG	Scanne et restaure la base de registre.
SET	Affiche, définit ou supprime des variables d'environnement Windows.
SETLOCAL	Commence la localisation des changements de l'environnement dans un fichier de commandes.
SETVER	Définit le numéro de version que MS-DOS fournit à un programme.
SHARE	Permet de verrouiller les fichiers lorsqu'ils peuvent être accédés par plusieurs programmes en même temps
SHIFT	Modifie la position des paramètres remplaçables dans un fichier de commandes.
SHUTDOWN	Éteint l'ordinateur.
SMARTDRV	Crée un cache disque dans la mémoire étendue.
SORT	Trie les éléments en entrée.
START	Lance une fenêtre pour l'exécution du programme ou de la commande.
SUBST	Affecte une lettre de lecteur à un chemin d'accès.
SWITCHES	Configure le clavier étendu.

SYS	Transférer les fichiers système vers un lecteur.
TELNET	Telnet vers un autre ordinateur / système.
TIME	Affiche ou définit l'heure de l'horloge interne du système.
TITLE	Définit le titre de la fenêtre MS-DOS.
TRACERT	Permet de voir le chemin parcouru entre votre poste et l'ordinateur qui héberge le site
TREE	Représente graphiquement l'arborescence d'un lecteur ou d'un chemin.
TYPE	Affiche le contenu d'un fichier texte.
UNDELETE	Récupère un fichier qui a été effacé.
UNFORMAT	Annule un formatage.
UNLOCK	Unlock un disque dur.
VER	Affiche le numéro de version.
VERIFY	Indique à Windows s'il doit ou non vérifier que les fichiers sont écrits correctement sur un disque donné.
VOL	Affiche le nom et le numéro de série du volume.
XCOPY	Copie des fichiers et des arborescences de répertoires.

Pour de plus ample détail sur chacune de ses commandes, consulter l'aide en ligne de commande. Pour cela, il suffit d'écran la commande exemple : DEL /?



Et l'aide s'affiche :

```
C:\Windows\system32\CMD.exe
Microsoft Windows [version 6.0.6002]
Copyright (c) 2006 Microsoft Corporation. Tous droits réservés.

C:\Users\Stéphane>DEL /?
Supprime un ou plusieurs fichiers.

DEL [/P] [/F] [/S] [/Q] [/A[[:]attributs]] noms
ERASE [/P] [/F] [/S] [/Q] [/A[[:]attributs]] noms

  noms          Spécifie une liste d'un ou plusieurs fichiers ou répertoires.
                Les caractères génériques peuvent être utilisés pour
                supprimer plusieurs fichiers. Si un répertoire est spécifié,
                tous les fichiers qu'il contient seront supprimés.

  /P            Demande une confirmation avant de supprimer un fichier.
  /F            Force la suppression de fichiers en lecture seule.
  /S            Supprime les fichiers dans tous les sous-répertoires.
  /Q            Pas de confirmation avant suppression avec car. génériques.
  /A            Suppression en fonction des attributs
  attributs     R  Fichiers en lecture seule     S  Fichiers système
                H  Fichiers cachés               A  Fichiers prêts pour archive
                I  Fichiers indexés sans contenu L  Points d'analyse
                -  Préfixe de négation

Si les extensions de commandes sont activées, DEL et ERASE sont modifiées
comme suit :
La logique d'affichage du commutateur /S est inversée. Elle
Appuyez sur une touche pour continuer...
```

4 – Les commandes spécifiques :

ECHO ON/OFF

Permet d'activer / désactiver l'affichage des commandes qui sont exécutées. Par défaut, c'est la commande **ECHO ON** qui est utilisée. L'utilisation de l'arobase permet de biffer le statut de la commande "**ECHO**". Par ailleurs, la commande "Echo off" évite l'affichage des commandes contenues dans le script.

```
@echo off
ECHO Bonjour
@echo on
```

ECHO message

Permet l'affichage du message. MS-DOS n'est pas sensible pour les commandes à la différence entre les majuscules et les minuscules, que vous écriviez echo ou Echo, ou bien encore ECHO ou EcHo, le résultat sera le même. N'utilisez pas d'accent, car MS-DOS va remplacer les caractères accentués par des symboles.

```
J'ai útú reþue ó mon examen !!!
```

ECHO.

Affiche une ligne blanche. Tilmo Salmi sur cette page web : www.uwasa.fi/~ts/http/http2.html#batch signale un problème concernant l'utilisation de la commande "**ECHO**" quand cela s'applique aux variables. Comparez la sortie-écran de ces deux fichiers Batch :

```
@echo off
SET variable=Ceci est un test
ECHO %variable%
SET variable=
ECHO %variable%
@echo on
```

Seul le second fichier Batch n'affiche pas d'erreur.

```
@echo off
SET variable=Ceci est un test
ECHO. %variable%
set variable=
ECHO. %variable%
@echo on
```

L'explication est simple : la commande "**ECHO**" suivie d'un point force l'affichage d'une ligne vide.

REM

Permet d'insérer des commentaires dans le fichier Batch. Les commentaires sont très importants dans les programmes.

```
@echo off
REM ceci est un commentaire dans un fichier .BAT
@echo on
```

La commande **REM** vous permet d'insérer des commentaires ou de désactiver temporairement des commandes incluses dans votre fichier batch. Ce n'est pas tout à fait vrai. La commande REM ECHO quelque chose > Sortie.txt ne fait qu'envoyer rien au fichier Sortie.txt. La commande est donc exécutée, mais en mode "désactivé". Une sorte de coup à blanc... Pour les amoureux de la performance il est donc plus judicieux d'utiliser :: plutôt que **REM**. Les deux points étant considérés comme l'indication d'une étiquette, la commande ne sera pas exécutée. À titre de test, saisissez tour à tour ces deux commandes :

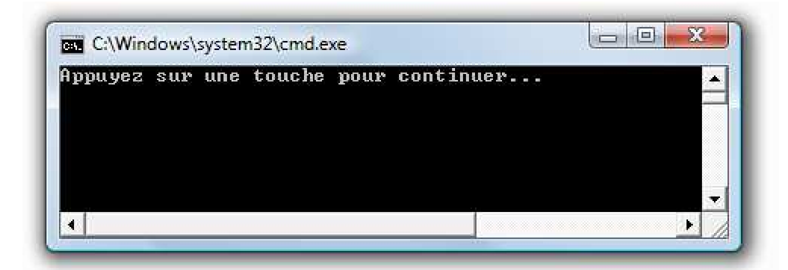
```
@echo off
REM ECHO A
:: ECHO B
@echo on
```

La seconde commande provoque un retour chariot et non un saut de ligne. Dans un fichier Config.sys il est possible de

désactiver une commande en utilisant un point-virgule : ;connexion=c:\dos\ramdrive.sys

PAUSE

Permet de suspendre l'exécution du fichier. Un message du style « Appuyer sur une touche pour continuer » s'affiche.



TITLE

Cette commande comme son nom l'indique change le titre par défaut : c'est-à-dire *cmd.exe*, par celui que vous définissez de cette façon : title suivi du titre que vous avez choisi

```
@echo off
TITLE Nettoyage
@echo on
```



COLOR

Celle-ci change la couleur de la police et celle de l'arrière-plan. Après *color* il faut indiquer une couleur en hexadécimal. Voici toutes les couleurs :

0 = Noir	8 = Gris
1 = Bleu foncé	9 = Bleu clair
2 = Vert	À = Vert clair
3 = Bleu gris	B = Cyan
4 = Marron	C = Rouge
5 = Pourpre	D = Rose
6 = Kaki	E = Jaune
7 = Gris clair	F = Blanc

Le premier caractère correspond à l'arrière-plan et le second à la police.

Par exemple :

```
@echo off
COLOR C5
ECHO TEST
PAUSE
@echo on
```



Ou

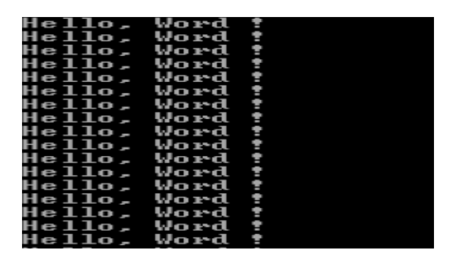
```
@echo off
COLOR 8F
ECHO TEST
PAUSE
@echo on
```



GOTO

Il s'agit d'un saut inconditionnel qui permet de casser la séquence d'exécution des commandes, de continuer l'exécution à un endroit donné différent dans le code. La commande "**GOTO**" permet d'aller au label du même nom. La marque ":Label" est un "mot" précédé de deux points (":"), correspondant à un repère utilisé par la commande **GOTO**. Un saut inconditionnel peut aussi être utilisé pour sauter des morceaux de code. Un saut inconditionnel peut être utilisé avec la commande "**IF**" pour exécuter ou ne pas exécuter du code en fonction d'une condition.

```
@echo off
:Start
echo Hello, Word !
Goto Start
```



Appuyer sur la combinaison des touches Ctrl + C : permet d'arrêter le programme en cours d'exécution.

La commande "**GOTO**" vous permet donc d'atteindre un point précis de votre script. La commande spécifiée doit porter un titre que nous appelons une étiquette. Cette ligne de titre commence obligatoirement par deux points. Nous sommes obligés d'anticiper quelque peu sur l'explication des conditions, car c'est une des utilisations privilégiées des étiquettes. Dans un nouveau fichier Batch copiez ce contenu :

```
@echo off
DIR *.doc
IF NOT errorlevel 1 GOTO Fin ELSE IF GOTO Avertissement
:Avertissement
ECHO Aucun fichier .doc !
:Fin
```

```
@echo on
```

La variable %Errolevel% nous permet de saisir si la commande "**DIR**" a renvoyé une valeur 1 ("Échec") ou 0 ("Opération réussie"). Si au moins un fichier .doc est trouvé, nous allons directement à l'étiquette :Fin. Ou, plus exactement si la sortie d'erreur n'est pas 1 ("Échec"), alors il faut se rendre à l'étiquette :Fin. Sinon (ELSE IF) nous nous rendrons à l'étiquette :Avertissement. Si nous rajoutons une commande à la suite de l'étiquette :Fin, cette dernière sera automatiquement exécutée même si la condition n'est pas remplie. Afin d'éviter cela, il nous faut nous servir d'une étiquette spéciale : goto:eof. Notre fichier script devient alors :

```
@echo off
DIR*.doc
IF NOT errorlevel 1 goto Fin ELSE IF GOTO Avertissement
:Avertissement
ECHO Aucun fichier .doc ! & goto:eof
:Fin
ECHO Processus finit !
@echo on
```

Dans ce dernier cas, la commande goto:eof nous permet d'effectuer une césure dans le script. Cela revient à dire : "Affiche le message 'Aucun fichier .doc !' puis ne fait plus rien". En termes savants, puisque nous ne définissons pas d'étiquette, nous transférons le contrôle à la fin du script en cours.

IF

Permet d'exécuter une commande si une condition est vérifiée. **IF** n'accepte qu'une seule commande à sa droite, c'est pour cela que la commande "**GOTO**" sera régulièrement utilisée, pour exécuter ou non certaines parties du Batch. Il y a différentes formes du **IF** : IF, IF EXIST, IF ERRORLEVEL et IF NOT qui peuvent être combinés. On retrouve les syntaxes suivantes :

if [not] errorlevel *Nombre Commande* [else *Expression*]

if [**not**] *Chaîne1*==*Chaîne2 Commande* [**else** *Expression*]

if [**not**] **exist** *NomFichier Commande* [**else** *Expression*]

Si les extensions de commandes sont activées, utilisez la syntaxe suivante :

if [**/i**] *Chaîne1 OpComparaison Chaîne2 Commande* [**else** *Expression*]

if cmdextversion *Nombre Commande* [**else** *Expression*]

if defined *Variable Commande* [**else** *Expression*]

D'où :

not	Spécifie que la commande doit être exécutée uniquement si la condition est fausse.
errorlevel *Nombre*	Spécifie que la condition est vraie uniquement si le programme précédent exécuté par Cmd.exe a renvoyé un code de sortie égal ou supérieur à *Nombre*.
Commande	Spécifie la commande qui doit être exécutée si la condition précédente est remplie.
Chaîne1==Chaîne2	Ne reconnaît que la condition est vraie que si *Chaîne1* et *Chaîne2* sont identiques. Ces valeurs peuvent être des chaînes littérales ou des variables de fichier de commandes (par exemple, *%1*). Vous n'avez pas besoin d'utiliser de guillemets autour des chaînes littérales.
exist *NomFichier*	Ne reconnaît la condition comme étant vraie que si *NomFichier* existe.
/i	Force les comparaisons de chaînes à ne pas tenir compte de la casse. Vous pouvez utiliser ***/i*** sur la forme *Chaîne1*==*Chaîne2* de **if**. Ces comparaisons sont génériques, en ce sens que si *Chaîne1* et *Chaîne2* sont tous deux constitués uniquement de chiffres numériques, les chaînes sont converties en nombres et une comparaison numérique est effectuée.
cmdextversion *Nombre*	Spécifie qu'une condition est vraie uniquement si le numéro de version interne associé à l'extension de commande de Cmd.exe est supérieur ou égal à *Nombre*. La première version est égale à 1. Elle est incrémentée d'une unité lorsque des améliorations significatives sont ajoutées aux extensions de commandes. L'expression conditionnelle **cmdextversion** n'est jamais vraie lorsque les extensions de commandes sont désactivées (par défaut, les extensions de commandes sont activées).
defined *Variable*	Ne reconnaît la condition comme étant vraie que si *Variable* est défini.
Expression	*Spécifie qu'une commande de ligne de commande et tous ses paramètres doivent être transmis à la commande dans une clause **else**.*
OpComparaison	Spécifie un opérateur de comparaison en trois lettres. Le tableau suivant répertorie les valeurs valides de *OpComparaison*.

Si les extensions de commandes sont activées, il est possible d'utiliser la syntaxe suivante :

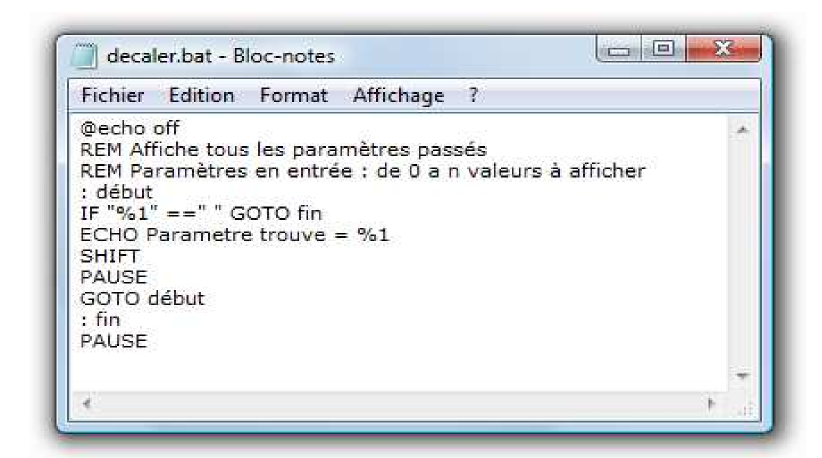
Supérieur ou égal à [/i] Chaîne1 Options_Comparaison Chaîne2 Commande [ELSE Expression]

Le commutateur **/i** permet de ne pas tenir compte de la casse. Tableau du récapitulatif les valeurs possibles pour les options de comparaison :

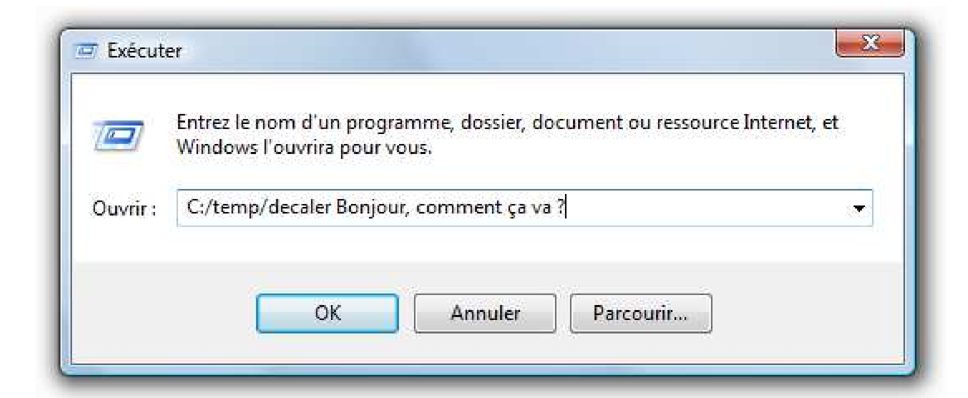
Opérateur	Description
EQU	Égal à
NEQ	Différent de
LSS	Inférieur à
LEQ	Inférieur ou égal à
GTR	Supérieur à
GEQ	Supérieur ou égal à

Prenons l'exemple suivant. On va afficher la liste des paramètres passés au programme Batch. Je crée un fichier que je nomme **decaler.bat** dans le répertoire C:\Temp.

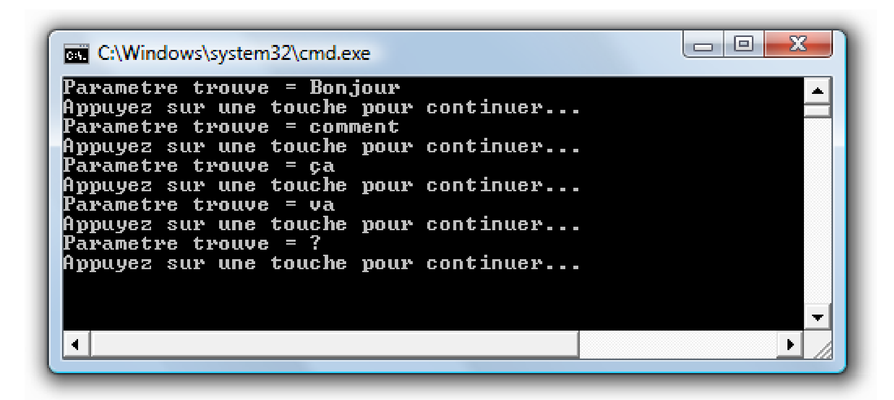
```
@echo off
REM Affiche tous les paramètres passés
REM Paramètres en entrée : de 0 a n valeurs à afficher
: début
IF "%1" ==" " GOTO fin
ECHO Parametre trouve = %1
SHIFT
PAUSE
GOTO début
: fin
PAUSE
```



On va le lancer de la manière suivante, à l'aide de la commande exécutée de Windows :



À chaque valeur reçue, on clique sur une touche pour obtenir les valeurs une à une :



Examinons cette commande :

IF cmdextversion Nombre Commande [else Expression]

Le mot clé cmdextversion permet de vérifier si les extensions de commandes sont activées ou non. La première version est égale à 1. Elle doit être indiquée dans "Nombre". Voici une autre possibilité :

IF defined Variable Commande [ELSE Expression]

Ne reconnaît la condition comme étant vraie que si "Variable" est défini. Un cas particulier concerne la vérification des répertoires. Si nous souhaitons nous assurer que le répertoire Test existe bien nous devrons saisir :

IF EXIST Test\nul (@echo "Le répertoire existe bien !") ELSE md Test

Cela revient à dire que si le répertoire Test existe bien un périphérique fictif (nul) existe aussi. Nous pouvons imaginer la création d'un fichier de commande qui se chargera de vérifier si un répertoire Test existe sur le disque dur. Si la commande précédente échoue (errorlevel 0) alors il sera créé. Copiez dans un nouveau fichier Batch ces trois lignes :

```
@echo off
DIR /s Test
IF errorlevel 0 (md Test)
```

Nous pourrions écrire la dernière ligne de cette façon :

IF %errorlevel% leq 0 (md C:\Test)

Dans ce dernier cas nous nous servons de errorlevel comme d'une variable qui sera égale ou non à 0. Nous pouvons aussi saisir ce type de commande :

IF NOT EXIST C:\Test\nul md C:\Test

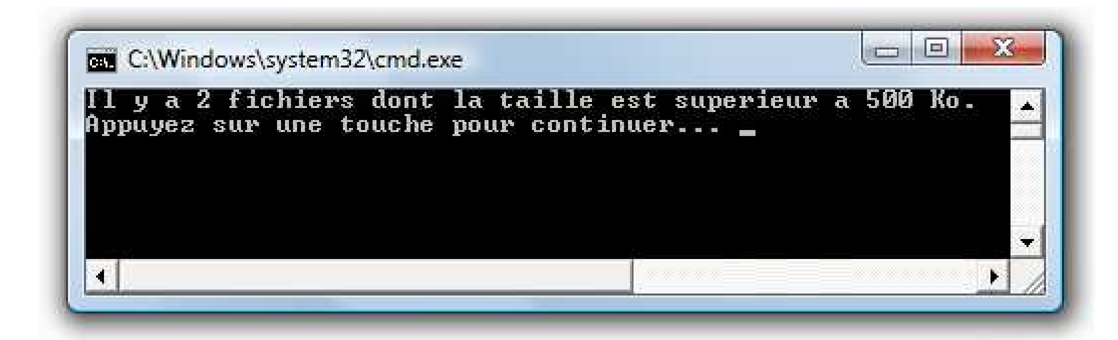
Vous pouvez conditionner la recherche à la condition que l'utilisateur saisisse le nom du fichier à rechercher :

```
@echo off
IF not exist %1 (echo Le fichier %1 est introuvable dans ce répertoire.) else echo Fichier trouvé !
```

La variable %1 représente dans ce cas la chaîne de caractères (le nom du fichier) que l'utilisateur aura saisie à la suite du nom du fichier de commande.

Prenons l'exemple suivant. On recherche dans le répertoire courant, tous les fichiers du type ***.jpg** qui ont une taille supérieure à 500 Ko (soir 500000 octets)

```
@echo off
REM Pour toutes les extensions mettre *.*
REM Ici on indique l'extension *.jpg
SET filtre=*.jpg
REM La taille est exprimée en octet
SET taille=500000
SET compteur=0
REM J'utilise l'option de comparaison /i GTR
FOR /r %%a in (%filtre%) DO IF /i %%~za GTR %taille% (
SET /a compteur+=1
)
ECHO Il y a %compteur% fichiers dont la taille est supérieure a 500 Ko.
PAUSE
@echo on
```



Pour lister les fichiers *.jpg supérieurs à 500 Ko on modifiera le code précèdent de la manière suivante :

```
@echo off
REM Pour toutes les extensions mettre *.*
REM Ici on indique l'extension *.jpg
SET filtre=*.jpg
REM La taille est exprimée en octet
SET taille=500000
SET compteur=0
ECHO Listing des fichiers *.jpg supérieur a 500 Ko.
ECHO.
REM J'utilise l'option de comparaison /i GTR
FOR /r %%a in (%filtre%) DO IF /i %%~za GTR %taille% (
SET /a compteur+=1
ECHO %%a
ECHO.
)
ECHO Il y a %compteur% fichiers dont la taille est supérieure a 500 Ko.
ECHO.
PAUSE
@echo on
```

Le signe == sert uniquement à tester une égalité.

SET

Permet d'afficher, de définir ou de supprimer des variables d'environnement. Utilisée sans paramètres, la commande **SET** affiche les paramètres d'environnement en cours.

variable = valeur : Permet d'affecter une valeur à une variable

Dans l'exemple suivant, on définit la variable fichier, on lui affecte une valeur (ici "C:\Temp\Test.xls") est on affiche sa valeur à l'écran.

```
@echo off
SET fichier="C:\Temp\Test.xls"
ECHO %fichier%
PAUSE
```



Dans l'exemple suivant, on demande à l'utilisateur de saisir son nom et on récupérer l'information dans la variable :

```
@echo off
SET /p nom=Votre nom:
REM On récupère le nom de l'utilisateur saisi dans la variable "nom"
```

```
ECHO bonjour %nom%
PAUSE
@echo on
```



À noter les sous-commandes suivantes :

/a	Permet d'indiquer pour le paramètre *chaîne* une expression numérique qui est évaluée.
/p	Affecte au paramètre *variable* la valeur d'une ligne d'entrées.

Utilisation de l'option **/a** : Le tableau suivant présente les opérateurs autorisés avec l'option **/a** par ordre de priorité décroissante.

Opérateur	**Opération effectuée**
< >	Groupement
* / % + -	Calcul
<< >>	Décalage logique
&	ET au niveau du bit
^	OU exclusif au niveau du bit
|	OU au niveau du bit
= *= /= %= += -= &= ^= |= <<= >>=	Attribution
,	Séparateur d'expression

EXIST/NOT EXIST nom de fichier

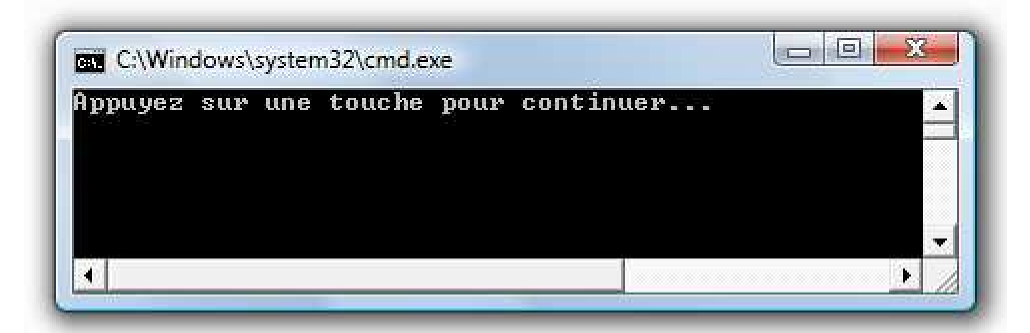
Permet de vérifier l'existence ou non d'un fichier. On test la présence du fichier « Test.xls » dans le répertoire courant, c'est-à-dire dans le répertoire où on exécute le fichier batch :

```
@echo off
IF EXIST Test.xls GOTO suite
REM S'il n'existe pas, on l'indique, s'il existe on passe à la suite
ECHO N'EXISTE PAS
PAUSE
: suite
REM Action de votre choix...
PAUSE
@echo on
```

Si le fichier « Test.xls » ne se trouve pas dans le répertoire courant :



Si le fichier « Test.xls » se trouve dans le répertoire courant :



Prenons un autre exemple, mais en utilisant un autre répertoire que celui où se situe notre fichier batch. Si un fichier nommé « **Test.doc** » se trouve dans le répertoire « C:\Temp » alors action de votre choix, sinon un message du style « Appuyer sur une touche pour continuer » s'affiche.

```
@echo off
SET fichier="C:\Temp\Test.xls"
IF EXIST %fichier% GOTO suite
REM S'il n'existe pas, on l'indique, s'il existe on passe à la suite
ECHO N'EXISTE PAS
PAUSE
: suite
REM Action de votre choix...
PAUSE
```

ERRORLEVEL/NOT ERRORLEVEL valeur

Chaque commande en fin d'exécution génère un code dont la valeur est comprise entre 0 et 255. Ce code est stocké dans la variable système **ERRORLEVEL**. On peut la tester après chaque exécution de commande la valeur de ce code avec cette commande. Le code 0 indique que la commande s'est bien déroulée.

CALL

Permet de lancer l'exécution d'un autre fichier Batch à partir d'un fichier Batch. Lorsque l'exécution du fichier Batch appelé est terminée, on revient à la commande suivante dans le fichier Batch appelant. La commande "**CALL**" vous permet donc d'appeler un programme sans que le programme "parent" soit stoppé. Créez un fichier de commandes nommé Test.bat contenant simplement ces lignes :

```
@echo off
DIR %1 /s> %2
@echo on
```

Pour envoyer le contenu d'un répertoire dans un fichier nommé Sortie.txt, saisissez :

```
@echo off
DIR %1 /s> %2
TEST . sortie.txt
@echo on
```

La première variable %1 est donc défini sur le répertoire par défaut (.) et la seconde (%2) sur le nom du fichier de sortie (Sortie.txt). La commande fonctionne quelque soit le répertoire par défaut et le nom du fichier de sortie. Nous aurions également pu saisir : test c:\Documents and settings\ sortie.txt

Créez un nouveau fichier de commande contenant ces lignes :

```
@echo Le nom du script est : %0
@echo Le nom de la variable 1 est : %1
@echo Le nom de la variable 2 est : %2
@echo Le nom de la variable: 3 est : %3
```

Enregistrez-le sous le même nom. Saisissez maintenant : test Jean Marc Michel. Créez maintenant un nouveau fichier de commande contenant ces lignes :

```
@echo off
Call :Chemin & goto:eof
```

```
:Chemin
@echo Voici le chemin: %~dp0
```

Ce script donne tout bêtement le chemin d'accès du répertoire courant. La commande "**CALL**" appelle une étiquette nommée :Chemin. La variable %~dp0 permet d'obtenir la lettre de lecteur et le chemin de l'arborescence visitée. L'appel vers l'étiquette :eof permet d'arrêter l'exécution du script une fois le chemin affiché. Vous pouvez placer ce script là où vous voulez puisqu'il aura toujours la capacité de retrouver le répertoire courant et les commandes qui seront placées dans la même arborescence. Il vous est possible de passer différentes variables si la commande principale en contient plusieurs.

Créez un fichier de commande (appelé A) contenant ces lignes :

```
@echo off
call b %1 %2
```

Et un second fichier (appelé B) contenant ceci :

```
@echo off
dir %1%\*.*
dir %2%\*.*
```

Saisissez cette commande : a c: c:\temp. Le fichier de commande A appelle le fichier B en lui passant les deux variables que vous avez saisies : %C:% et %C:\Temp%. Par ailleurs, il est possible de se servir des commandes "**CALL**" et "**GOTO**" pour créer des sous-routines. Vérifiez la sortie-écran produite par ce "Batch" :

```
@echo off
ECHO commande1
CALL :echo2
ECHO commande4
GOTO :eof
: echo2
ECHO commande2
CALL :echo3
GOTO :eof
: echo3
ECHO commande3
GOTO :eof
```

L'avantage de la commande "**CALL**" sur un simple appel d'étiquette induit par la commande "**GOTO**" est que le déroulement du script n'est pas interrompu. Nous appelons simplement les instructions contenues sous l'étiquette puis reprenons le déroulement de notre "Batch" là où nous l'avions laissé.

COPY

Copie un ou plusieurs fichiers à partir d'un emplacement dans un autre.

```
@echo off
REM Copie des dossiers et sous-dossiers de C:\Temp\A vers C:\Temp\B
REM paramètre en entrée : dossier source et dossier destination
SET origine="C:\Temp\A"
SET destination="C:\Temp\B"
ECHO copie de %origine% vers %destination%
REM Si le répertoire existe on ne le crée pas
IF EXIST %destination% GOTO suite
ECHO Création nouveau dossier de destination
MD %destination%
GOTO copie
: suite
ECHO Dossier de destination existe déjà
: copie
REM Copie les répertoires et sous-répertoire (même vides)
COPY %origine% %destination%
@echo on
```

Dans le deuxième exemple, on ne copie que les fichiers portant l'extension « **.txt** » de « **C:\Temp\A** » vers « **C:\Temp\B** ».

```
@echo off
REM Sauvegarder les fichiers txt du répertoire C:\Temp\A, s'il y en a
REM vers C:\Temp\B
SET origine="C:\Temp\A"
SET destination="C:\Temp\B"
```

```
REM Tester l'existence de fichiers TXT dans le répertoire C:\Temp\A
IF NOT EXIST %origine% /v .txt GOTO etiqRIEN
ECHO Il y a des documents a sauvegarder
REM Copier avec vérification
COPY %origine%\*.txt %destination% /v
goto fin
:etiqRien
ECHO Rien à copier
: fin
PAUSE
@echo on
```

À noter les sous-commandes suivantes :

/d	Permet, à l'emplacement de destination, d'enregistrer sous une forme décryptée les fichiers cryptés qui subissent l'opération de copie.
/v	Vérifie que les nouveaux fichiers sont correctement écrits.
/n	Utilise un nom de fichier court, éventuellement disponible, lors de la copie d'un fichier dont le nom comporte plus de huit caractères ou dont l'extension compte plus de trois caractères.
/y	Supprime la confirmation de remplacement d'un fichier de destination existant.
/-y	Invite l'utilisateur à confirmer le remplacement d'un fichier de destination existant.
/z	Copie des fichiers mis en réseau en mode redémarrage.
/a	Désigne un fichier texte ASCII.
/b	Désigne un fichier binaire.

Dans l'exemple suivant, on copie le fichier "A.txt" du répertoire courant vers le dossier "C:\Temp\B". Si le dossier de destination "B" n'existe pas, celui-ci est alors créé. Puis le fichier "A.txt" est copié dans le dossier de réception "B".

```
@echo off
REM Copie le fichier A.txt vers le répertoire C:\Temp\B
REM paramètres en entrée : dossier source et dossier destination
SET origine="A.txt"
SET destination="C:\Temp\B"
ECHO copie de %origine% vers %destination%
REM Si le répertoire existe on ne le crée pas
IF EXIST %destination% GOTO suite
ECHO Création nouveau dossier de destination
MD %destination%
GOTO copie
: suite
ECHO Dossier de destination existe déjà
: copie
REM Copie le fichier et en assure la copie intégrale
COPY /v %origine% %destination%
@echo on
```

XCOPY

Copie des fichiers et des répertoires, sous-répertoires compris.

```
@echo off
REM Copie des dossiers et sous-dossiers de C:\Temp\A vers C:\Temp\B
REM Paramètres en entrée : dossier source et dossier destination
SET origine="C:\Temp\A"
SET destination="C:\Temp\B"
ECHO Copie de %origine% vers %destination%
REM Si le répertoire existe on ne le crée pas
IF EXIST %destination% GOTO suite
ECHO Création nouveau dossier de destination
MD %destination%
GOTO copie
: suite
ECHO Dossier de destination existe déjà
: copie
REM Copie les répertoires et sous-répertoire (même vides)
XCOPY %origine% %destination% /e
```

```
REM Attribut en lecture seule les fichiers, dossiers et sous-dossiers
ATTRIB %destination% +r /d /s
PAUSE
@echo on
```

Dans le deuxième exemple, on ne copie que les fichiers portant l'extension « **.txt** » de « **C:\Temp\A** » vers « **C:\Temp\B** » en incluant les sous-répertoires :

```
@echo off
REM Sauvegarder les fichiers txt du répertoire C:\Temp\A, s'il y en a
REM vers C:\Temp\B
SET origine="C:\Temp\A"
SET destination="C:\Temp\B"
REM Tester l'existence de fichiers TXT dans le répertoire C:\Temp\A
IF NOT EXIST %origine% /v .txt GOTO etiqRIEN
ECHO Il y a des documents a sauvegarder
REM Copier avec vérification
XCOPY %origine%\*.txt %destination% /e /v
goto fin
:etiqRien
ECHO Rien à copier
: fin
PAUSE
@echo on
```

À noter les sous-commandes suivantes :

/w	Affiche le message suivant et attend une réponse avant de commencer à copier les fichiers : appuyez sur une touche pour lancer la copie des fichiers.
/p	Demande à l'utilisateur de confirmer la création de chaque fichier de destination.
/c	Ne tient pas compte des erreurs.
/v	Vérifie chaque fichier au fur et à mesure de son écriture dans le fichier de destination afin de garantir que les fichiers de destination sont identiques aux fichiers sources.
/q	Supprime l'affichage des messages de la commande xcopy.
/f	Affiche les noms des fichiers sources et de destination pendant la copie.
/l	Affiche la liste des fichiers à copier.
/g	Crée des fichiers de destination décryptés.
/d[:*mm-jj-aaaa*]	Copie uniquement les fichiers sources qui ont été modifiés à la date précisée ou après cette date. Si vous n'utilisez pas le paramètre *mm-jj-aaaa*, la commande xcopy copie tous les fichiers *Source* qui sont plus récents que les fichiers de *Destination* existants. Cette option de ligne de commandes permet de ne mettre à jour que les fichiers qui ont été modifiés.
/u	Copie uniquement les fichiers sources qui existent dans la destination.
/i	Si le paramètre *Source* correspond à un répertoire ou qu'il contient des caractères génériques et que le paramètre *Destination* n'est pas spécifié, la commande xcopy suppose que le nom du répertoire de destination est *destination* et un nouveau répertoire est donc créé. Tous les fichiers sont ensuite copiés dans ce nouveau répertoire. Par défaut, la commande xcopy demande à l'utilisateur si la destination est un fichier ou un répertoire.
/s	À moins qu'ils ne soient vides, copie les répertoires et les sous-répertoires. En l'absence de cette option de ligne de commandes, xcopy travaille dans un seul répertoire.
/e	Copie tous les sous-répertoires, même s'ils sont vides. Utilisez /e en combinaison avec les options de ligne de commandes /s et /t.
/t	Copie uniquement la structure (l'arborescence) du sous-répertoire et non les fichiers. Pour copier des répertoires vides, vous devez inclure l'option de ligne de commandes /e.
/k	Copie les fichiers et conserve l'attribut de lecture seule pour les fichiers de destination si les fichiers sources possèdent cet attribut. Par défaut, xcopy supprime l'attribut de lecture seule.
/r	Copie les fichiers avec un attribut de lecture seule.
/h	Copie les fichiers dotés des attributs fichier caché et fichier système. Par défaut, la commande xcopy ne copie pas les fichiers cachés ou les fichiers système.

/a	Copie uniquement les fichiers sources dotés de l'attribut archive. Cette option de ligne de commandes ne permet pas de modifier l'attribut archive du fichier source. Pour plus d'informations sur l'activation de cet attribut à l'aide de la commande attrib, consultez Rubriques connexes.
/m	Copie les fichiers sources dotés de l'attribut archive. Contrairement à l'option de ligne de commandes /a, l'option /m permet de désactiver l'attribut archive des fichiers spécifiés dans la source. Pour plus d'informations sur l'activation de cet attribut à l'aide de la commande attrib, consultez Rubriques connexes.
/n	Crée des copies en utilisant les noms de fichier ou de répertoire abrégés NTFS. Cette option de ligne de commandes est requise lors de la copie de fichiers ou de répertoires d'un volume NTFS vers un volume FAT ou lorsque les conventions de noms de fichier du système de fichiers FAT (8.3) sont exigées sur le système de fichiers de destination. Le système de fichiers de destination peut être le système FAT ou NTFS.
/o	Copie les informations relatives à l'appartenance des fichiers et à la liste de contrôle d'accès discrétionnaire (DACL, Discretionary access control list)
/x	Copie les paramètres d'audit des fichiers et les informations relatives à la liste de contrôle d'accès du système (SACL, System access control list) (implique l'utilisation de l'option /o).
/exclude:*NomFichier1*[+[*NomFichier2*]][+[*NomFichier3*]]	Spécifie la liste des fichiers contenant des chaînes.
/y	Supprime la demande de confirmation de remplacement d'un fichier de destination existant.
/-y	Affiche la demande de confirmation de remplacement d'un fichier de destination existant.
/z	Permet la copie sur un réseau en mode redémarrage.
/?	Affiche l'aide à l'invite de commandes.

DEL (erase)

Supprime les fichiers spécifiés

```
@echo off
REM Supprime le fichier Test.doc du répertoire C:\Temp
DEL "C:\Temp\Test.doc"
REM Idem que précédemment
ERASE "C:\Temp\Test2.doc"
REM Supprime tous les fichiers ainsi que ceux du sous-répertoire du répertoire spécifié
REM Demande la confirmation de suppression sur chaque répertoire et sous répertoires
REM Affiche le nom des fichiers au fur et à mesure qu'ils sont supprimés.
DEL "C:\Temp\A" /s
REM Supprime tous les fichiers ainsi que ceux du sous-répertoire du répertoire spécifié
DEL "C:\Temp\B" /s /q
@echo on
```

Dans le deuxième exemple, on ne supprimer que les fichiers portant l'extension « **.txt** » du répertoire et des sous-répertoires de « **C:\Temp\B** » :

```
@echo off
SET repertoire="C:\Temp\B"
REM Supprime tous les fichiers *.txt ainsi que ceux du sous-répertoire du répertoire spécifié
DEL %repertoire%\*.txt  /s /q
REM En inscrivant PAUSE, j'aurai la liste des fichiers supprimés
PAUSE
@echo on
```

PS : Si vous souhaitez confirmer la suppression de chaque fichiers .txt trouvés utilisez la commande suivante : DEL %repertoire%*.txt /p/s/q

Dans le troisième exemple, on supprime tous les fichiers nommés « **Folder.ini** », « **AlbumSmall.jpg** » et « **Folder.jpg** » du répertoire et des sous-répertoires de « **C:\Temp\B** » :

```
@echo off
SET repertoire="C:\Temp\B"
DEL %repertoire%\Folder.ini  /s /q /a /f
```

```
DEL %repertoire%\AlbumSmall.jpg  /s /q /a /f
DEL %repertoire%\Folder.jpg  /s /q /a /f
PAUSE
@echo on
```

À noter les sous-commandes suivantes :

/p	Vous invite à confirmer que vous voulez supprimer le fichier spécifié.
/f	Force la suppression des fichiers dotés de l'attribut lecture seule.
/s	Supprime les fichiers spécifiés du répertoire en cours et de tous ses sous-répertoires. Affiche le nom des fichiers au fur et à mesure qu'ils sont supprimés.
/q	Spécifie le mode silencieux. Vous n'êtes pas invité à confirmer l'ordre de suppression.
/a	Supprime les fichiers dotés des attributs spécifiés.

DIR

Affiche la liste des fichiers et des sous-répertoires d'un répertoire. Utilisée sans paramètre, la commande **DIR** affiche le nom et le numéro de série du volume du disque, suivis de la liste des répertoires et des fichiers du disque, y compris leur nom ainsi que la date et l'heure de leur dernière modification. Pour les fichiers, **DIR** affiche l'extension du nom ainsi que la taille en octets. Il affiche également le nombre total des fichiers et des répertoires de la liste, leur taille cumulée et l'espace restant (en octets) sur le disque. L'exemple suivant liste l'intégralité des fichiers du répertoire C:\Temp ainsi que tous ses sous-répertoires :

```
@echo off
DIR "C:\Temp" /s /b
PAUSE
@echo on
```

À noter les sous-commandes suivantes :

/p	Affiche un écran de la liste à la fois. Pour faire apparaître l'écran suivant, appuyez sur n'importe quelle touche du clavier.
/s	Affiche toutes les apparitions du nom de fichier spécifié dans le répertoire désigné et tous ses sous-répertoires.
/q	Affiche les informations relatives à la propriété du fichier.
/b	Présente chaque nom de répertoire ou de fichier (extension comprise) sur une ligne séparée. **/b** n'affiche pas d'informations d'en-tête ou de résumé. **/b** annule **/w**.
/l	Affiche les noms de répertoire et de fichier en minuscules sans les trier. Le paramètre **/l** ne convertit cependant pas les caractères étendus en minuscules.
/c	Affiche le séparateur des milliers pour les tailles de fichier.
/q	Affiche les informations relatives à la propriété du fichier.
/a [[:] ***attributs*]**	N'affiche que les noms des répertoires et fichiers dotés des attributs spécifiés. En l'absence de **/a**, **dir** affiche les noms de tous les fichiers, à l'exclusion des fichiers cachés et systèmes. Si vous utilisez **/a** sans spécifier *attributs*, **dir** présente les noms de tous les fichiers, fichiers cachés et système compris. La liste ci-dessous décrit chacune des valeurs susceptibles d'être utilisées avec le paramètre *attribut*. Le signe deux-points (:) est facultatif. Employez n'importe quelle combinaison de ces valeurs sans les séparer par des espaces :

Valeur	Description
H	Fichiers cachés
s.	Fichiers système
d	Répertoires
a	Fichiers prêts pour l'archivage
r	Fichiers lecture seule
-h	Fichiers qui ne sont pas cachés
-s	Fichiers autres que les fichiers système
-d	Fichiers uniquement (sans répertoires)
-a	Fichiers qui n'ont pas été modifiés depuis la dernière sauvegarde
-r	Fichiers qui ne sont pas en lecture seule

/o [[:]*OrdreTri*]	N'affiche que les noms des répertoires et fichiers dotés des attributs spécifiés. En l'absence de **/a**, **dir** affiche les noms de tous les fichiers, à l'exclusion des fichiers cachés et systèmes. Si vous utilisez **/a** sans spécifier *attributs*, **dir** présente les noms de tous les fichiers, fichiers cachés et système compris. La liste ci-dessous décrit chacune des valeurs susceptibles d'être utilisées avec le paramètre *attribut*. Le signe deux-points (:) est facultatif. Employez n'importe quelle combinaison de ces valeurs sans les séparer par des espaces : détermine l'ordre de tri et d'affichage des noms de répertoire et de fichier utilisés par **dir**. En l'absence de **/o**, **dir** affiche les noms dans l'ordre selon lequel ils se présentent dans le répertoire. Si vous utilisez **/o** sans préciser *OrdreTri*, **dir** affiche les noms des répertoires par ordre alphabétique, puis les noms des fichiers, également par ordre alphabétique. Le signe deux-points (:) est facultatif. La liste suivante décrit chacune des valeurs susceptibles d'être assignées au paramètre *OrdreTri*. Employez n'importe quelle combinaison de ces valeurs sans les séparer par des espaces :

Valeur	Description
n	Ordre alphabétique des noms
e	Ordre alphabétique des extensions
d	Ordre chronologique des dates et des heures, à commencer par la plus ancienne
s.	Ordre de taille, à commencer par la plus petite
g	Répertoires groupés avant les fichiers
-n	Ordre alphabétique inverse des noms (de Z à A)
-e	Ordre alphabétique inverse des extensions (de .ZZZ à .AAA)
-d	Ordre chronologique des dates et des heures, à commencer par la plus récente
-s	Ordre de taille, à commencer par la plus grande
-g	Répertoires groupés à la suite des fichiers

Ici, on reprend l'exemple en y mettant un ordre de tri par croissant (en commençant par le plus gros fichier) :

```
@echo off
DIR "C:\Temp" /s /b /o:-s
PAUSE
@echo on
```

/t [[:]*ChampHeure*]	Spécifie le champ d'heure à afficher ou à utiliser pour le tri : La liste ci-dessous décrit chacune des valeurs susceptibles d'être utilisées avec le paramètre *ChampHeure*.

Valeur	Description
c	Création
a	Accès précédent
w	Enregistrement précédent

Autres exemples :

Pour afficher la liste de tous les noms de fichier suivis de l'extension .txt stockés dans les répertoires du lecteur C:\Temp, tapez :

```
@echo off
DIR "C:\Temp\*.txt" /w /o /s /p
PAUSE
@echo on
```

```
C:\Windows\system32\cmd.exe
 Le volume dans le lecteur C s'appelle BOOT
 Le numéro de série du volume est 4EF5-7986

 Répertoire de C:\Temp

debug.txt     Login.txt     WMALog.txt
               3 fichier(s)           33 647 octets

 Répertoire de C:\Temp\Requêteur

Extensions.txt
Historique.txt
Listing Des Chemins + Fichiers.txt
Listing Des Dossiers.txt
Listing Des Fichiers (Sans Extension).txt
Listing Des Fichiers.txt
Parametres.txt
               7 fichier(s)          286 383 octets

     Total des fichiers listés :
              10 fichier(s)          320 030 octets
               0 Rép(s)  97 732 313 088 octets libres
Appuyez sur une touche pour continuer...
```

Autre exemple : On liste les fichiers avec l'extension *.**jpg** dans le répertoire courant (C'est à dire là où se trouve votre fichier Batch)

```
@echo off
DIR *.jpg /s /b
PAUSE
@echo on
```

S'il n'y a aucun fichier avec l'extension *.**jpg** dans votre répertoire courant, vous aurez un message d'erreur :



Afin d'éviter ce genre de message, on utilise la commande IF :

```
@echo off
IF EXIST *.jpg GOTO suite
REM S'il n'existe pas, on l'indique et on quitte le programme, s'il existe on passe à la suite
ECHO N'EXISTE PAS
ECHO.
PAUSE
EXIT
: suite
REM Action de votre choix... On affiche le listing des fichiers *.jpg trouvés
DIR *.jpg /s /b
ECHO.
PAUSE
@echo on
```

```
C:\Windows\system32\cmd.exe
C:\Temp\Test\Test 1.jpg
C:\Temp\Test\Test 2.jpg
C:\Temp\Test\Test 3.jpg

Appuyez sur une touche pour continuer...
```

RENAME (REN)

Modifie le nom d'un fichier ou d'un groupe de fichiers.

```
@echo off
REN "C:\Temp\flash10.log" flash10.txt
PAUSE
@echo on
```

Supposez que vous souhaitiez modifier les extensions de tous les noms de fichier du répertoire en cours qui portent l'extension .txt en remplaçant par exemple, les extensions .txt par des extensions .doc. Pour cela, tapez :

```
@echo off
REN "C:\Temp\*.txt" *.doc
PAUSE
@echo on
```

Pour renommer un répertoire nommé A en B, situé dans C:\Temp tapez :

```
@echo off
REN "C:\Temp\A" B
PAUSE
@echo on
```

Pour renommer toutes les extensions *.doc du répertoire courant en *.txt :

```
@echo off
REN *.doc *.txt
PAUSE
@echo on
```

Idem pour convertir en minuscule toutes les extensions du répertoire courant *.MP3 en *.mp3

```
@echo off
REN *.MP3 *.mp3
PAUSE
@echo on
```

Pour renommer toutes les extensions des fichiers *.mp3 (en minuscule) par *.MP3 (en majuscule) dans le répertoire courant et les sous-répertoires !

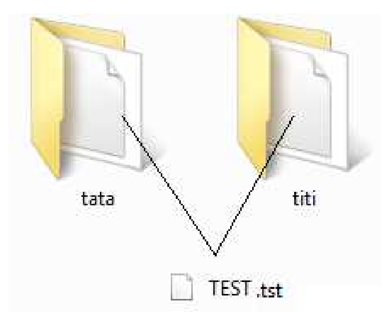
```
@echo off
for /r %%i in (*.mp3) do ren "%%i" "*.MP3"
PAUSE
@echo on
```

Pour renommer des fichiers commençant par AA par BBBB dans le répertoire courant :

```
@echo off
for %%i in (AA*) do ren "%%i" "XX%%i"
ren "XXAA*" "BBBB*"
PAUSE
@echo on
```

Renommer un fichier avec le nom du répertoire courant : Pour notre exemple, on suppose la structure suivante : Le dossier

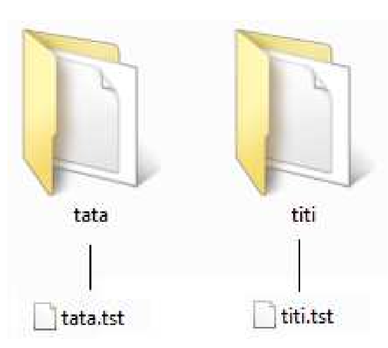
« Toto » contient deux dossiers : « Titi » et « Tata ». Chaque dossier contient un fichier « TEST.tst ».



Le batch suivant renomme les fichiers « TEST.tst » de chacun des dossiers par leur nom réceptif.

```
@echo OFF
CD
FOR /D %%A IN (*) DO (
CD %%A
RENAME TEST.tst "%%A.tst"
CD ..
)
PAUSE
@echo on
```





RMDIR (ou MD)

Cette commande permet de supprimer un répertoire.

```
@echo off
REM Pour supprimer un dossier
RMDIR C:\Temp\A
REM Pour supprimer un dossier et tous les sous-dossiers
RMDIR C:\Temp\B /s
REM Pour supprimer un dossier et tous les sous-dossiers sans demander de confirmation
RMDIR C:\Temp\C /s /q
PAUSE
@echo on
```

Voici la liste des sous-commandes :

/s	Supprime le répertoire spécifié et tous ses sous-répertoires, y compris tous les fichiers qui y figurent. Ce commutateur est utilisé pour supprimer une arborescence.
/q	Exécute la commande **rmdir** en mode silencieux. Supprime les répertoires sans confirmation.

MKDIR (ou MD)

Crée un répertoire ou sous-répertoire.

```
@echo off
REM Pour créer un nouveau dossier
MKDIR C:\Temp\A
REM Pour créer un nouveau dossier et un sous-dossier
MKDIR C:\Temp\B\C
@echo on
```

Lorsque vous activez les extensions de commande (ce qui est le cas par défaut), vous pouvez utiliser une seule commande mkdir pour créer des répertoires intermédiaires dans un chemin spécifié.

```
@echo off
MKDIR C:\Taxes\Propriété\Courant
@echo on
```

Cela équivaut à taper la suite de commandes ci-dessous quand les extensions de commande sont désactivées :

```
@echo off
MKDIR C:\Taxes
CHDIR C:\Taxes
MKDIR Propriété
CHDIR Propriété
MKDIR Courant
@echo on
```

CHDIR (CD)

Affiche le nom du répertoire en cours ou modifie le dossier en cours. Utilisée seulement avec une lettre de lecteur (notamment, **chdir** C:), **CHDIR** affiche les noms du lecteur et du dossier en cours. Utilisée sans paramètres, la commande

➔ Lister les dossiers contenus dans un dossier :

Voilà un fichier batch qui liste les dossiers contenus dans un dossier parent. Prenons l'exemple d'un dossier nommé « Test ». Ce dossier contient 3 sous-dossiers : 1, 2 et 3.



On trouve dans notre dossier « Test », un fichier nommé « Test.bat ». Voici sa composition :

```
@echo off
CD
FOR /D %%r IN (*) DO (
ECHO %%r
)
PAUSE
@echo on
```

Et voice le résultat :



CHDIR affiche le lecteur et le répertoire en cours.

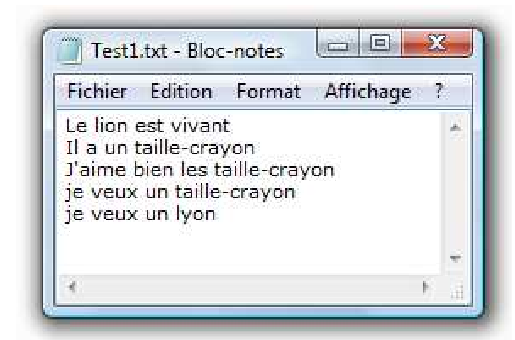
PRINT

Imprime un fichier texte.

```
@echo off
PRINT "C:\Temp\B\flash10.txt"
@echo on
```

FIND

Cherche une chaîne de texte déterminée dans un ou plusieurs fichiers. Après avoir cherché la chaîne dans le(s) fichier(s) spécifié(s), **FIND** affiche toutes les lignes de texte qui contiennent la chaîne spécifiée. Voici un fichier texte nommé Test1.txt sous C:\Temp :



Pour afficher toutes les lignes du fichier Test1.txt qui contiennent la chaîne « taille-crayon », tapez :

```
@echo off
find "taille-crayon" "C:\Temp\Test1.txt"
PAUSE
@echo on
```

Voici le résultat :



L'exemple suivant affiche tous les fichiers « **.txt** » du répertoire « C:\Temp » :

```
@echo off
DIR "C:\Temp" /s /b | find ".txt"
PAUSE
@echo on
```

Voici la liste des sous-commandes :

/v	Affiche toutes les lignes qui ne contiennent pas la *chaîne* spécifiée.
/c	Compte les lignes qui contiennent la *chaîne* spécifiée et affiche le total.
/n	Fait précéder chaque ligne du fichier par son numéro.
/i	Spécifie que la recherche ignore la casse.

Pour trouver une chaîne qui contient du texte entre guillemets, vous devez d'abord mettre toute la chaîne entre guillemets. Deuxièmement, vous devez utiliser deux guillemets pour chaque guillemet contenu dans la chaîne. Pour trouver, "Les chercheurs ont écrit dans leur rapport la mention "version préliminaire." Il ne s'agit pas de la version définitive du rapport." dans Rapport.doc, tapez :

```
@echo off
find "Les chercheurs ont écrit dans leur rapport la mention ""version préliminaire." Il ne s'agit pas de la version définitive du rapport." "C:\Temp\Rapport.doc"
PAUSE
@echo on
```

Pour chercher un jeu de fichiers, utilisez la commande **FIND** avec la commande **FOR**. Pour rechercher dans le répertoire courant les fichiers qui possèdent l'extension .bat et qui contiennent la chaîne "INVITE", tapez :

```
@echo off
FOR %f in (*.bat) do FIND "INVITE" %f
PAUSE
@echo on
```

Pour rechercher dans votre disque dur et afficher les noms de fichier du lecteur C qui contiennent la chaîne "UC", utilisez le signe | pour diriger les résultats d'une commande **DIR** vers **FIND**, comme suit :

```
@echo off
DIR c:\ /s /b | FIND "UC"
PAUSE
@echo on
```

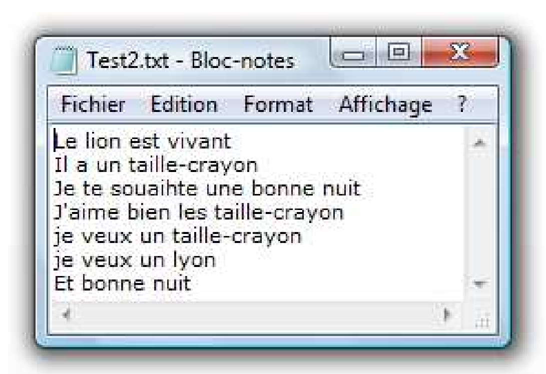
Comme **FIND** distingue les majuscules des minuscules lors de la recherche et que **DIR** produit des données de sortie en majuscules, tapez la chaîne "UC" en majuscules ou utilisez l'option de ligne de commande **/i** avec **FIND**.

FINDSTR

Recherche de structures de texte dans des fichiers à l'aide d'expressions régulières. Utilisez des espaces pour séparer plusieurs chaînes de rechercher sauf si l'argument est précédé de /c. Pour rechercher "bonne" ou "nuit" dans le fichier Test2.txt, tapez :

```
@echo off
FINDSTR "bonne nuit" "C:\Temp\Test2.txt"
```

```
PAUSE
@echo on
```



Voici la liste des sous-commandes :

/b	Ne reconnaît un modèle qu'au début d'une ligne.
/e	Ne reconnaît un modèle qu'à la fin d'une ligne.
/l	Utilise les chaînes de recherche littéralement.
/r	Utilise les chaînes de recherche comme expressions régulières. **Findstr** interprète tous les métacaractères comme des expressions régulières sauf si vous utilisez **/l**.
/s	Recherche les fichiers concordants dans le répertoire en cours ainsi que dans tous ses sous-répertoires.
/p	Omet les fichiers contenant des caractères non imprimables.
/f:*fichier*	Lit la liste des fichiers à partir du fichier spécifié.
/c:*chaîne*	Utilise le texte spécifié comme chaîne de recherche littérale.
/g:*fichier*	Obtient les chaînes de recherche à partir du fichier spécifié.
/d:*liste_répertoire*	Effectue la recherche dans une liste de répertoires séparés par des virgules.
/a:*AttributCouleur*	Spécifie des attributs de couleur avec deux chiffres hexadécimaux.

Autres exemples :

Pour trouver toutes les occurrences du mot Windows (avec la majuscule W) dans le fichier Devis.txt, tapez :

findstr Windows devis.txt

Pour chercher le mot Windows, sans distinguer la casse, dans tous les fichiers du répertoire en cours et de tous les sous-répertoires, tapez :

findstr /s /i Windows *.*

Pour trouver toutes les occurrences des lignes contenant le mot STOP, précédé par un nombre quelconque d'espaces (comme dans une boucle de programme informatique, par exemple) et pour inclure le numéro de ligne où se trouve chaque occurrence trouvée, tapez :

findstr /b /n /c:" *STOP" *.bas

Si vous souhaitez chercher différents éléments dans le même jeu de fichiers, créez un fichier texte contenant chaque critère de recherche sur une nouvelle ligne. Vous pouvez également indiquer les fichiers exacts que vous souhaitez rechercher dans un fichier texte. Pour utiliser les critères de recherche dans le fichier Trouver.txt, rechercher les fichiers indiqués dans ListeFich.txt, puis stocker les résultats dans le fichier Result.fin, tapez :

findstr /g:finddata.txt /f:filelist.txt > results.out

Supposons que vous souhaitiez trouver tous les fichiers du répertoire en cours et tous les sous-répertoires contenant le mot ordinateur, sans distinction de la casse. Pour obtenir la liste de tous les fichiers contenant le mot ordinateur, tapez :

findstr /s /i /m "\<ordinateur\>" *.*

Supposons maintenant que vous souhaitiez trouver non seulement le mot "ordinateur", mais également n'importe quel mot

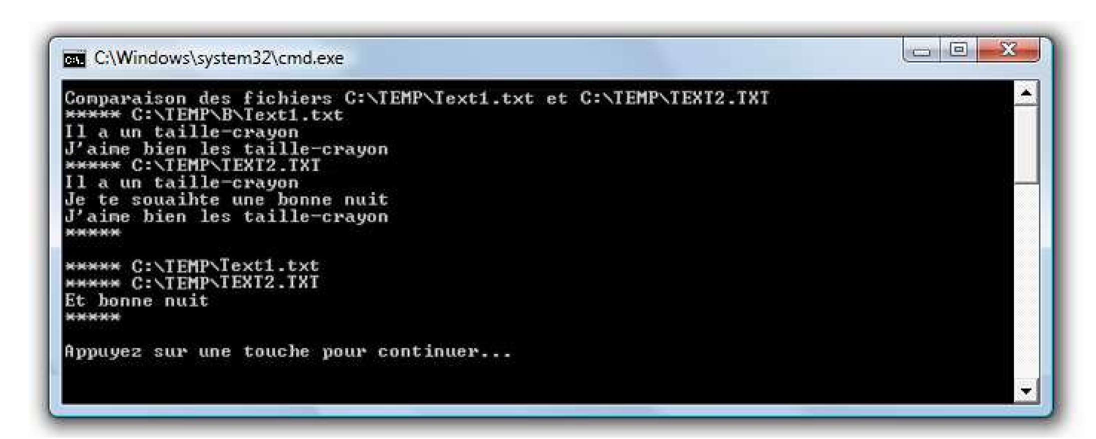
commençant par les lettres ordi, tel qu'"ordinal" et "ordinaire". ; comme suit :

```
findstr /s /i /m "\<ordi.*" *.*
```

FC

Compare deux fichiers et affiche leurs différences. Pour effectuer une comparaison ASCII de deux fichiers de texte nommés Test1.txt et Test2.txt et afficher les résultats en format abrégé, tapez :

```
@echo off
FC /a "C:\Temp\Test1.txt" "C:\Temp\Test2.txt"
PAUSE
@echo on
```



Voici la liste des sous-commandes :

/a	Abrège le résultat d'une comparaison ASCII. Au lieu d'afficher toutes les lignes différentes, **fc** n'affiche que la première et la dernière ligne de chaque jeu de différences.
/b	Compare les fichiers en mode binaire. **Fc** compare les deux fichiers octet par octet et ne tente pas de resynchroniser le fichier après la découverte d'une discordance. C'est le mode de comparaison par défaut des fichiers qui possèdent les extensions de fichier suivantes : .exe, .com, .sys, .obj, .lib ou .bin.
/c	Ne distingue pas la casse.
/l	Compare les fichiers en mode ASCII. **Fc** compare les deux fichiers ligne par ligne et tente de resynchroniser les fichiers après la découverte d'une discordance. C'est le mode de comparaison par défaut des fichiers, à l'exception de ceux qui possèdent les extensions de fichier suivantes : .exe, .com, .sys, .obj, .lib ou .bin.
/lb*n*	Spécifie le nombre *n* de lignes de la zone tampon de lignes interne. Le nombre de lignes par défaut de cette zone s'élève à 100. Si le nombre de lignes consécutives différentes des fichiers que vous comparez est supérieur à ce nombre, **fc** annule la comparaison.
/n	Affiche les numéros des lignes au cours d'une comparaison ASCII.
/t	Empêche **fc** de convertir les tabulations en espaces. Le comportement par défaut consiste à traiter les tabulations comme s'il s'agissait d'espaces, avec des taquets tous les huit caractères.
/u	Compare les fichiers comme s'il s'agissait de fichiers texte Unicode.
/w	Compresse les espaces blancs (c'est-à-dire, les tabulations et les espaces) au cours de la comparaison. Si une ligne contient un grand nombre de tabulations ou d'espaces consécutifs, **/w** traite ces caractères comme un seul espace. Utilisée avec l'option de ligne de commande **/w**, **fc** ne tient pas compte (et n'effectue pas de comparaison) des espaces blancs au début et à la fin d'une ligne.
/*nnnn*	Indique le nombre de lignes consécutives qui doivent correspondre l'une à l'autre pour que **fc** considère que les fichiers doivent être resynchronisés. Si le nombre de lignes correspondantes des fichiers est inférieur à *nnnn*, **fc** affiche ces lignes en tant que différences. La valeur par défaut est 2.

Autres Exemples :

Pour effectuer une comparaison binaire de deux fichiers de commandes nommés Profits.bat et Benefice.bat, tapez :

FC /b profits.bat benefice.bat

Pour comparer chaque fichier .bat du répertoire en cours au fichier Nouv.bat, tapez :

FC *.bat nouv.bat

Pour comparer le fichier Nouv.bat du lecteur C au fichier Nouv.bat du lecteur D, tapez :

FC c:nouv.bat d:*.bat

Pour comparer chaque fichier de commandes du répertoire racine du lecteur C au fichier du même nom du répertoire racine du lecteur D, tapez :

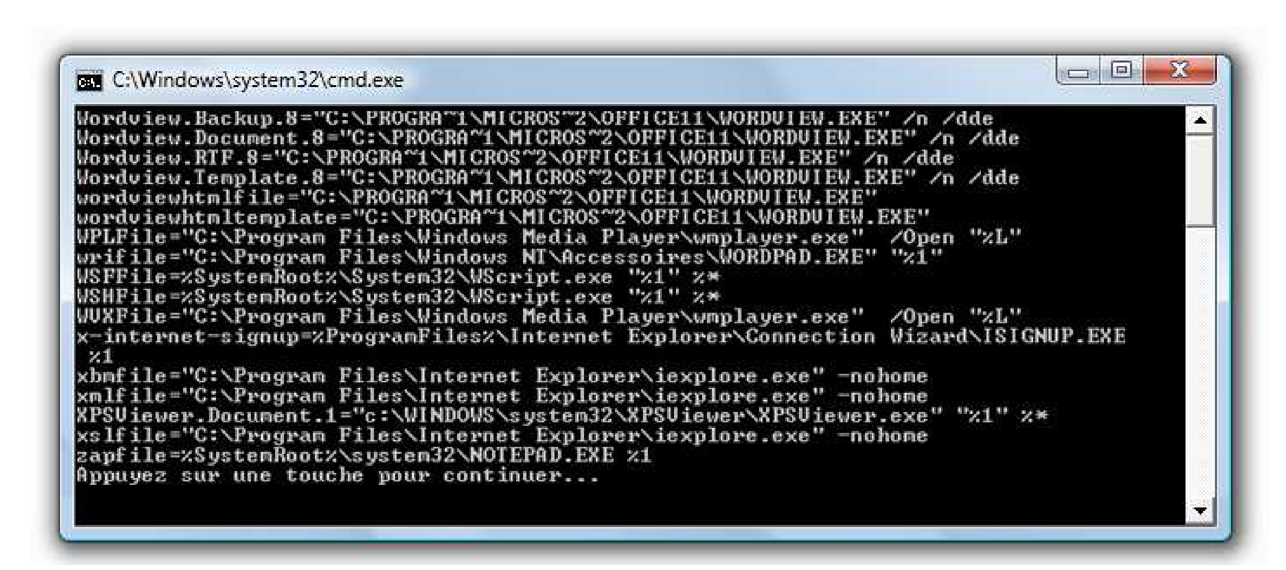
FC c:*.bat d:*.bat

FTYPE

Affiche ou modifie les types des fichiers employés dans les associations d'extensions de fichiers. Utilisé sans paramètres, **FTYPE** affiche les types de fichiers possédant des chaînes de commandes d'ouverture définies.

Pour afficher les types de fichiers actuels possédant des chaînes de commandes d'ouverture définies, tapez :

```
@echo off
FTYPE
@echo on
```



Autres exemples :

Pour afficher la chaîne de commande d'ouverture en cours pour un type de fichier spécifique, tapez :

ftype *TypeFichier*

Pour supprimer la chaîne de commande d'ouverture d'un type de fichier spécifique, tapez :

ftype *TypeFichier*=Type :ASSOC .pl=PerlScript FTYPE PerlScript=perl.exe %1 %*

Pour appeler le script Perl, tapez :

script.pl 1 2 3

Pour ne plus avoir à taper les extensions, tapez :

set PATHEXT=.pl;%PATHEXT%

FTP

Transfère des fichiers vers et depuis un ordinateur exécutant un service de serveur FTP (File Transfer Protocol) comme les Services Internet (IIS, *Internet Information Services*). La commande **FTP** peut être utilisée de façon interactive ou en mode par lot en traitant des fichiers texte ASCII. Se rapprocher de l'aide de Windows.

```
@echo off
FTP puweb01
@echo on
```

```
C:\Windows\system32\cmd.exe
C:\Documents and Settings\PB04341\Mes documents\Pro BIP\Infothèqude
uweb01                                                          /dde
Connecté à cuweb01.PROBTP.
220 CUWEB01 FTP server (Version 4.2 Tue Jun 24 11:26:40 CDT 2008 /dde
Utilisateur (cuweb01.PROBTP:(none)) :
```

START

Ouvre une fenêtre d'invite de commandes distincte pour exécuter une commande ou un programme déterminé. Utilisée sans paramètres, la commande **START** ouvre une seconde fenêtre d'invite de commandes.

```
@echo off
REM Ouvre l'application Paint
START C:\WINDOWS\system32\mspaint.exe
REM Ouvre l'application Paint en pleine écran
START  /max C:\WINDOWS\system32\mspaint.exe
REM Ouvre l'image Infotheque.jpg du répertoire Temp dans Paint en pleine écran
START  /max C:\WINDOWS\system32\mspaint.exe /Temp\Infotheque.jpg
@echo on
```

Voici la liste des sous-commandes :

min	Démarre une nouvelle fenêtre d'invite de commandes réduite à une icône.
max	Démarre une nouvelle fenêtre d'invite de commandes en plein écran.
wait	Démarre une application et attend qu'elle se termine.

```
@echo off
REM Même programme que ci-dessus, mais en ouvrant les applications une par une
REM Ouvre d'abord l'application Paint
START/wait C:\WINDOWS\system32\mspaint.exe
REM Ouvre ensuite l'application Paint en pleine écran
START /wait /max C:\WINDOWS\system32\mspaint.exe
REM Pour finir, ouvre l'image Infotheque.jpg du répertoire Temp dans Paint en pleine écran
START /wait /max C:\WINDOWS\system32\mspaint.exe /Temp\Infotheque.jpg
ECHO Voilà la difference...
PAUSE
@echo on
```

Ci-dessous, vous trouvez quelques exemples supplémentaires permettant d'ouvrir et d'exécuter des programmes à l'aide ou non de la commande START.

<u>Ouvrir un fichier Excel :</u>

```
@echo off
Start /max Test.xls
@echo on
```

<u>Lancer un fichier mp3</u> :

```
@echo off
REM Pour ouvrir une mp3 il suffit d'écrire son nom complet
C:\Documents\MP3\Titre.mp3
@echo on
```

Et votre lecteur mp3 se lance automatiquement ! Si par exemple vous avez un lecteur mp3 qui peut créer une playlist comme jetaudio "playliste.pls", vous pourrez ouvrir plusieurs chansons mp3 ou autres, il vous suffit de créer votre playlist et de l'enregistrer. Ensuite, configurez votre fichier bat en écrivant simplement le nom complet comme par exemple :

```
@echo off
REM Pour ouvrir une playlist (ici avec le lecteur jetaudio)
C:\Documents\MP3\Playlist.pls
@echo on
```

Et votre lecteur ce lance automatiquement ...

Ouvrir un dossier avec Explorer.exe :

```
@echo off
explorer.exe,/e,/n,/root,%SystemRoot%
@echo on
```

CACLS

Affiche ou modifie les fichiers DACL (Discretionary Access Control List).

```
@echo off
CACLS "C:\MSOCache" /T /G "Tout le monde":f system:f administrateurs:f
REM A noter XCACLS, la même commande, mais en anglais
XCACLS "C:\MSOCache" /T /G "Tout le monde":f system:f administrateurs:f
PAUSE
@echo on
```

Voici la liste des sous-commandes :

/t	Modifie les listes DACL des fichiers spécifiés dans le répertoire en cours ainsi que tous ses sous-répertoires.
/e	Modifie une liste DACL au lieu de la remplacer.
/c	Continue de modifier les listes DACL, en ignorant les erreurs.
/g *Utilisateur*:*autorisation*	Accorde des droits d'accès à l'utilisateur spécifié.
/r *Utilisateur*	Révoque les droits d'accès de l'utilisateur spécifié.
/p *Utilisateur*:*autorisation*	Remplace les droits d'accès de l'utilisateur spécifié.
/d	Affiche de l'aide à l'invite de commandes.

Le tableau suivant répertorie les valeurs admises pour *autorisation*.

Valeur	Description
n	Aucune
r	Lecture
w	Écriture
c	Changer (écrire)
f	Contrôle total

4 – <u>Passage de paramètres</u> :

Il est possible de paramétrer un fichier Batch. On peut ainsi l'exécuter en insérant des données externes. Les paramètres dans un fichier Batch sont définis sous la forme %numéro (%1, %2...). Chaque paramètre est identifié par son numéro qui correspond à l'ordre dans lequel on transmettra ces paramètres lors de l'exécution du fichier Batch. Le paramètre %0 est toujours réservé au nom du fichier Batch en cours d'exécution.

Il n'est possible d'utiliser que 10 paramètres au maximum. Pour pallier ce problème, on a recours à la commande **SHIFT** qui

permet de décaler les paramètres d'un écran (le paramètre 2 devient paramètre 1, le paramètre 3 devient paramètre2...)

SHIFT

Permet de changer la position des paramètres de commandes dans un fichier de commandes.

SHUTDOWN

Cette commande permet d'éteindre ou fermer son ordinateur. Elle permet soit :

- D'éteindre son PC directement ou après *x* secondes
- De redémarrer son PC directement ou après *x* secondes
- De fermer la session avec les mêmes options…

C:\Windows\system32\CMD.exe

```
C:\Users\Stéphane>SHUTDOWN /?
Utilisation : SHUTDOWN [/i | /l | /s | /r | /g | /a | /p | /h | /e] [/f]
    [/m \\ordinateur][/t xxx][/d [p|u:]xx:yy [/c "commentaire"]]

    Sans argument    Afficher l'aide. Cela revient à entrer /?.
    /?          Afficher l'aide. Cela revient à n'entrer aucune option.
    /i          Afficher l'interface utilisateur graphique (GUI).
                Ce doit être la première option.
    /l          Fermer la session. Ne peut pas être utilisé avec l'option /m
                ou /d.
    /s          Arrêter l'ordinateur.
    /r          Arrêter et redémarrer l'ordinateur.
    /g          Éteignez et redémarrez l'ordinateur. Une fois le système
                réinitialisé, redémarrez toute application enregistrée.
    /a          Annuler un arrêt du système.
                Cela peut être utilisé uniquement pendant la période de délai.
    /p          Arrêter l'ordinateur local sans délai d'expiration ou
                avertissement.
                Peut être utilisé avec l'option /d ou /f.
    /h          Mettre l'ordinateur local en veille prolongée.
                Utilisable avec l'option /f.
    /e          Documenter la raison de l'arrêt inattendu d'un ordinateur.
    /m \\ordinateur Spécifier l'ordinateur cible.
    /t xxx      Définir la période de délai avant l'arrêt au bout de xxx secondes
.
                La plage valide est comprise entre 0 et 600, 30 étant la
                valeur par défaut.
                L'utilisation de /t xxx implique l'option /f.
    /c "commentaire" Commentaire sur la raison du redémarrage ou de l'arrêt.
                512 caractères maximum autorisés.
    /f          Forcer la fermeture des applications en cours d'exécution sans av
ertir les utilisateurs.
                /f est automatiquement défini lors d'une utilisation en conjoncti
on avec /t xxx.
    /d [p|u:]xx:yy  Fournir la raison du redémarrage ou de l'arrêt.
                p indique que le redémarrage ou l'arrêt est planifié.
                u indique que la raison est définie par l'utilisateur.
                  Si ni p ni u ne sont spécifiés, le redémarrage ou l'arrêt n'est
 pas planifié.
                xx est le code de raison majeur (entier positif inférieure à 256)
.
                yy est le code de raison mineur (entier positif inférieur à 65536
).
```

```
C:\Windows\system32\CMD.exe

Raisons sur cet ordinateur :
(E = Attendu U = Inattendu P = planifié, C = défini par le client)
Type     Majeur  Mineur  Titre

 U       0       0       Autre (non planifié)
E        0       0       Autre (non planifié)
E P      0       0       Autre (planifié)
 U       0       5       Autre panne : le système ne répond pas
E        1       1       Matériel : maintenance (non planifiée)
E P      1       1       Matériel : maintenance (planifiée)
E        1       2       Matériel : installation (non planifiée)
E P      1       2       Matériel : installation (planifiée)
  P      2       3       Système d'exploitation : mise à jour (planifiée)
E        2       4       Système d'exploitation : reconfiguration (non planifiée)

E P      2       4       Système d'exploitation : reconfiguration (planifiée)
  P      2       16      Système d'exploitation : Service pack (Planifié)
         2       17      Système d'exploitation : correctif logiciel (non planifi
é)
  P      2       17      Système d'exploitation : correctif logiciel (planifié)
         2       18      Système d'exploitation : correctif de sécurité (Non plan
ifié)
  P      2       18      Système d'exploitation : correctif de sécurité (Planifié
)
E        4       1       Application : maintenance (non planifiée)
E P      4       1       Application : maintenance (planifiée)
E P      4       2       Application : installation (planifiée)
E        4       5       Application : aucune réponse
E        4       6       Application : instable
 U       5       15      Échec du système : erreur d'arrêt
E        5       19      Problème de sécurité
 U       5       19      Problème de sécurité
E P      5       19      Problème de sécurité
E        5       20      Perte de connexion réseau (Non planifié)
 U       6       11      Panne d'alimentation : cordon déconnecté
 U       6       12      Panne d'alimentation : environnement
  P      7       0       Arrêt d'une interface API héritée
```

5 – <u>Les Boucles</u> :

Après avoir fait connaissance avec une technique de la programmation des sauts inconditionnels (**GOTO**), en voici une autre. Nous allons créer un petit batch qui va afficher successivement les chiffres 1 à 4. Écrivez le fichier batch suivant :

```
@echo off
FOR %%A in (1 2 3 4) Do Echo C'est le nombre %%A
PAUSE
@echo on
```



Ce fichier Batch contient une boucle **FOR...DO**. À quoi sert-elle ? Tout d'abord, %%A est utilisé seulement en tant que nom de variable. Cette variable prend alors toutes les valeurs de la liste spécifiée entre les parenthèses : Dans notre cas, %%A prend donc successivement les valeurs 1, 2, 3, et 4. Les valeurs constituant la liste doivent être séparées entre elles par des espaces, des virgules, ou des points-virgules. Ensuite, la commande qui suit immédiatement est exécutée avec la valeur prise par la variable %%A. Dans notre cas, on verra à l'écran le message "C'est le nombre" suivi de la valeur de la variable à chaque exécution de **ECHO**. Un autre intérêt de cette commande est que les éléments de la liste peuvent être des noms de fichiers. Ainsi il est possible d'exécuter une seule commande pour plusieurs fichiers. Vous pouvez donc afficher à l'écran plusieurs fichiers à la fois avec une seule commande qui est TYPE :

```
FOR %%A IN (AUTOEXEC.BAT CONFIG.SYS) DO TYPE %%A
```

Vous pouvez aussi utiliser les caractères génériques, par exemple :

FOR %%A IN (*.TXT *.BAT) DO TYPE %%A

Tous les fichiers texte et Batch s'afficheront à l'écran. Une boucle **FOR... DO...** permet d'utiliser une variable prenant successivement toutes les valeurs d'une liste prédéfinie, et l'utilisation de cette variable dans des commandes DOS ou Batch standard.

Créer une boucle :

Dans un nouveau fichier Batch copiez ce contenu :

```
@echo off
SET variable=
: Boucle
set /a variable+=1
IF /i %variable% equ 10 GOTO :eof
ECHO %variable% && GOTO Boucle
```

Nous posons une variable sans la définir. Pour vous en rendre compte saisissez ces commandes :

set variable=
set variable

Nous créons une étiquette nommée : Boucle qui va effectuer les actions suivantes :

set /a variable+=1 : Incrémenter la variable de départ d'un pas de 1. De 0 notre variable va donc être égale à 1.
if /i %variable% equ 11 goto :eof : Si la variable est égale au nombre 10 arrêter l'exécution du script.

Dans le cas contraire, inscrire à l'écran la valeur obtenue puis reprendre l'exécution du fichier de commandes à partir de l'étiquette nommée :Boucle. Si nous voulons obtenir un pas de "deux" il suffit de modifier la ligne SET /a variable+=1 par ceci : SET /a variable+=2

Créer un compteur :

Créez un fichier Batch avec ce contenu :

```
@echo off
SET filtre=*.*
SET taille=0
SET compteur=0
FOR /r %%a in (%filtre%) DO IF %%~za==%taille% (
SET /a compteur+=1
)
ECHO Il y a %compteur% fichiers Dont la taille est de 0 octet.
PAUSE
@echo on
```

À l'aide de la commande "**SET**" nous définissons trois variables :

filtre : Dans ce cas la recherche portera sur tous les fichiers *.*.
taille : Définit la taille du fichier à chercher. Dans notre cas : 0 ko.
compteur : Qui est tout d'abord défini à 0 puis incrémenté à chaque fois de 1 (SET /a compteur+=1) dès qu'un nouveau fichier répondant aux critères définis a été signalé.

La commande FOR /r démarre une recherche récursive jusqu'à ce que plus aucun nouveau fichier ne soit trouvé. Chaque nom de fichier est passé sous forme de variable (%a) qui se réinitialise à chaque nouveau fichier trouvé et le lot de commandes exécuté. Le commutateur %~za développe %a jusqu'à la taille du fichier. L'utilisation des parenthèses nous permet de définir que si les critères de recherche sont remplis alors la variable %compteur% est incrémenté d'une unité. Le commutateur /a permet d'indiquer que c'est une expression numérique qui sera évaluée pour la variable %compteur%. La dernière ligne affiche simplement la dernière valeur de la variable %compteur%.

Afficher les fichiers textes un par un et les comptabiliser :

Afficher tous les noms des fichiers textes du répertoire courant et des sous-répertoires un par un :

```
@echo off
REM Affiche tous les noms des fichiers textes du répertoire et des sous-répertoires un par un
SET compteur=0
```

```
FOR /r %%a IN (*.txt) DO (@ECHO %%a
PAUSE
SET /a compteur+=1
)
ECHO %compteur% fichier(s) texte(s) trouve(s)...
PAUSE
@echo on
```

6 – Aller plus loin dans les fichiers Batch :

Manipuler les variables en Invite de commandes :

L'Invite de commandes a prévu un certain nombre de variables internes :

Nom de la variable	Fonction
%CD%	Affiche le répertoire courant.
%DATE%	Affiche la date.
%TIME%	Affiche l'heure.
%RANDOM%	Affiche une valeur décimale comprise entre 0 et 32767.
%ERRORLEVEL%	Affiche le code d'erreur renvoyée par la précédente commande.
%CMDEXTVERSION%	Affiche la version du processeur de commande.
%CMDCMDLINE%	Affiche la ligne de commande originelle invoquée par le processeur de commande.

La commande **SET p** affiche toutes les variables commençant par la lettre P. Une suite possible de commandes est :

```
set variable=c:\test
set variable
```

Nous pouvons vérifier que notre variable est bien égale à C:\Répertoire. Nous pouvons également saisir :

```
DIR %variable%
```

Le contenu de l'arborescence C:\Test s'affichera. Une autre manière consiste à saisir le nom de l'arborescence à la suite du nom du fichier de commande. Par exemple :

```
batch c:
```

Le script nommé Batch permettant de lister les fichiers texte du répertoire indiqué aura alors ce contenu :

```
DIR %1%\*.txt
```

Nous pouvons procéder à l'extraction de tous les caractères à l'exception des trois premiers. Saisissez donc :

```
SET 1=DIR *.txt
ECHO %1:~3%
ECHO %1:~-3%
```

Seuls les trois derniers caractères seront affichés.

```
ECHO %1:~0,-4%
```

Tous les caractères seront affichés à l'exception des 4 derniers.

```
ECHO %1:~0,3%
```

Nous afficherons les trois premiers caractères de la variable passée précédemment.

```
ECHO %1:~4,5%
```

Dans ce dernier cas, 5 caractères seront affichés en partant du 4ème caractère. Attention : Si vous utilisez un caractère réservé dans votre variable, ce dernier doit être "signalé". Afin de passer une variable nommée Test&1 il vous faudra saisir :

```
SET x =test^&1
```

Si vous ne faites pas précéder le caractère réservé du signe ^ vous aurez droit à un message d'erreur salé ! Par exemple, saisissez ces commandes :

```
SET x=%time%
SET y=%date%
ECHO %x% %y%
ECHO %x:~0,5% %y%
```

Nous reprenons deux variables internes pour définir deux nouvelles variables nommées x et y. Il est possible de cacher tous les répertoires en saisissant cette commande :

```
SET dircmd=0
```

Le contenu des arborescences restera invisible bien que leur accès reste possible. Afin de revenir à la situation précédente, saisissez :

```
SET dircmd=
```

Astuce : Il est possible d'assigner à une variable un nombre aléatoire. Saisissez ces commandes :

```
SET variable=%random%
ECHO %variable%
```

"Transtyper" une variable :

Il est donc possible de modifier une variable en additionnant à sa définition toute sorte de chaînes de caractères. Cette opération s'appelle du "transtypage". Par exemple, saisissez ces commandes :

```
SET datation =La date d'aujourd'hui est le %date%
SET datation
```

Créez un nouveau fichier Batch contenant :

```
@echo off
SET chaîne=[Ceci n'est qu'un test]
SET chaîne=%chaîne:[=%
SET chaîne=%chaîne:]=%
ECHO %chaîne%
@echo on
```

Dans cet exemple, nous ne faisons que dire qu'aux caractères [et] rien ne correspond (et donc ils ne s'afficheront pas).
Le script suivant affichera : (Ceci n'est qu'un test)

```
@echo off
SET chaîne=[Ceci n'est qu'un test]
SET chaîne=%chaîne:[=(%
SET chaîne=%chaîne:]=)%
ECHO %chaîne%
@echo on
```

Nous pouvons remplacer la deuxième et troisième ligne par :

```
SET chaîne=%chaîne:~1,21%
```

Cela appelle deux remarques :

* La numérotation commence à partir de zéro : au caractère [correspond donc le numéro 0.
* Les espaces sont comptés.

Afin de convertir une chaîne de caractères de minuscules en majuscules :

```
@echo off
SET chaîne=%1
SET chaîne=%chaîne:a=A%
SET chaîne=%chaîne:b=B%
SET chaîne=%chaîne:c=C%
::Vous pouvez indiquer toutes les lettres de l'alphabet...
SET chaîne=%chaîne:y=Y%
SET chaîne=%chaîne:z=Z%
ECHO %chaîne%
@echo on
```

Nous passons dans un premier temps une variable nommée %chaîne% qui sera la chaîne de caractère saisie par l'utilisateur ou, si nous préférons, le premier paramètre saisi à la suite du nom du fichier Batch : %1. Le reste du script opère la substitution des variables en changeant à chaque fois la casse de la lettre.

Créez un formulaire permettant de récupérer une ou plusieurs variables :

En Invite de commandes, la solution est assez simple... Saisissez ces commandes :

```
SET /p Variable=Veuillez entrer votre nom:
SET variable
```

Il est donc possible de se servir de "**SET**" en lieu et place de "**CHOICE**". Créez un nouveau fichier Batch en copiant les lignes suivantes :

```
@echo off
:début
ECHO 1 : Menu1
ECHO 2 : Menu2
ECHO 3 : Menu3
SET /p choix=choisissez un chiffre.
IF not %choix%=='' SET choix=%choix:~0,1%
IF %choix%==1 goto Menu1
IF %choix%==2 goto Menu2
IF %choix%==3 goto Menu3
ECHO %choix% n'est pas bon !
GOTO début
:Menu1
ECHO Bonjour & GOTO:eof
:Menu2
ECHO Au revoir & GOTO:eof
:Menu3
ECHO Adieu & GOTO:eof
```

Les quatre premières lignes affichent un formulaire. La quatrième ligne nous permet d'attendre que l'utilisateur ait saisi la chaîne qui sera associée à la variable %choix% que nous allons définir. Dans ce cas le commutateur /p est obligatoire. Si l'utilisateur appuie sur la touche Entrée sans avoir saisi de chiffre, la variable %choix% sera défini sur le chiffre 2. Pour vous en rendre compte, saisissez ces commandes :

```
SET choix=%choix:~0,1%
ECHO %choix%
```

Dans ce cas, nous nous rendons directement à l'étiquette n°2 : Menu2. Si le chiffre saisi n'est ni 1, 2 ou 3 alors il sera affiché un message d'erreur puis nous retournerons en début de script à l'étiquette :début. Chaque menu affiche un message puis termine l'exécution du script par la commande GOTO:eof.

Nous savons donc qu'il est possible de spécifier plusieurs conditions "à la chaîne". Il suffit de dire que si aucune des trois premières conditions n'a été remplie alors c'est que la saisie de l'utilisateur n'est pas valable. C'est la seule petite astuce de ce script !

Faire des calculs en Invite de commandes :

Une possibilité consiste à utiliser la commande "Set". Saisissez ceci :

```
SET /a z=356/356/1*356
```

Le tableau suivant présente les opérateurs autorisés avec le commutateur /a par ordre de priorité décroissante :

Opérateur	**Opération effectuée**
< >	Groupement
* / % + -	Calcul
<< >>	Décalage logique
& ET	au niveau du bit
^ OU	exclusif au niveau du bit
| OU	au niveau du bit

= *= /= %= += -= &= ^= |= <<= >>=	Attribution
,	Séparateur d'expression

Toute chaîne non numérique est traitée comme un nom de variable dont les valeurs sont converties en nombre avant d'être utilisées. Si une variable est traitée sans être définie dans l'environnement en cours, sa valeur sera égale à zéro. L'ajout du signe % est donc inutile. Il est facile d'imaginer une variable qui soit celle-ci : 0010. Nous pouvons supprimer les zéros placés devant en saisissant cette commande :

SET /a variable=100%variable%%100

Avec l'opérateur % est effectuée une division modulo. Dans une division modulo, les deux valeurs sont divisées, mais le résultat n'est que le reste de la division. Si vous saisissez : 19 % 5, vous obtenez comme résultat modulo 4, parce que 19 divisés par 5 égal 3 reste 4. Afin de le vérifier, saisissez ces commandes :

SET /a variable=19
SET /a variable=%variable%%5

Par ailleurs, si vous saisissez cette commande :

SET /a variable=0010

Vous obtenez comme résultat le chiffre 8. En fait, les nombres sont en notation décimale à moins qu'ils ne soient préfixés par 0x pour les valeurs hexadécimales ou par 0 pour la notation octale. Il est donc possible d'obtenir instantanément la valeur décimale d'un chiffre hexadécimal. À titre de test, saisissez ceci : set /a 0x3e7. Ce type de commande marche : (Set /a 75 /15) > sortie.txt

Voici un exemple de script tout simple :

```
@echo off
CALL :fonction %1 %2
ECHO La somme est %somme%
CALL :soustraction %1 %2
ECHO La soustraction est %soustraction%
GOTO :eof
: fonction
SET /a somme=%1+%2
: soustraction
SET /a soustraction=%1-%2
```

Saisissez le nom de votre fichier Batch suivi des deux éléments de l'opération. Nous récupérons les deux paramètres en nous servant des variables %1 et %2. La commande "SET /a" nous permet de calculer la valeur respective des variables %somme% et %soustraction%.

Justifier un nombre inclus dans une variable :

Dans un nouveau fichier batch copiez ce contenu :

```
@echo off
SET nombre=%1
IF %1 lss 10000 set a=
IF %1 lss 1000 set a=0
IF %1 lss 100 set a=00
IF %1 lss 10 set a=000
SET nombre=%a%%nombre%%
SET nombre=%nombre:"=%
ECHO %nombre%
```

Saisissez le nom de votre fichier batch suivi du nombre à justifier. Nous récupérons le paramètre saisi par l'utilisateur en vérifiant si le nombre et inférieur à 10000 puis à 1000 et ainsi de suite. À chaque fois nous définissons une variable nommée %a% que nous ferons précédées à la variable %nombre%. L'avant-dernière ligne nous permet simplement de supprimer les guillemets de la variable obtenue.

Localiser les variables d'environnement :

Les modifications d'environnement définies par la commande "Setlocal" ne sont appliquées localement qu'à votre fichier de commandes. Lorsqu'une commande "Endlocal" est rencontrée, les paramètres précédents sont rétablis. Il n'est pas possible d'utiliser ces variables indépendamment d'un script ou d'un fichier de commandes. Il nous est ainsi possible d'appeler un fichier de commandes situé à un emplacement différent de celui à partir duquel nous lançons le premier fichier de

commandes. Par exemple, créez un fichier nommé **A.bat** contenant ces lignes :

```
@echo off
setlocal
path=F:\;%path%
CALL B >c:\sortie.txt
endlocal
START notepad c:\sortie.txt
```

Créez un autre fichier nommé **B.bat** que vous placerez sur un autre lecteur (dans cet exemple F:) et qui contiendra la commande à exécuter : DIR *.txt. Lancez le fichier **A.bat**... La commande path=F:\;%path% nous permet de préciser que la variable définie %F:% vient s'ajouter aux autres chemins prédéfinis par la variable d'environnement %path%. La commande CALL B appelle à partir du fichier **A.bat** le fichier **B.bat.** cela revient à lancer l'exécution d'un second script à partir d'un premier script. Si nous n'avions pas ajouté le lecteur F: dans le **"**Path", le fichier **B.bat** n'aurai pas été trouvé. Les commandes "Setlocal" et "Endlocal" ne sont là que pour éviter un éventuel conflit entre la définition d'un même nom de variable dans un même script ou entre deux scripts exécutés au cours de la même session d'Invite de commandes.

Définir différentes variables dans un fichier Batch :

Imaginons que nous souhaitons pouvoir exécuter trois commandes différentes en fonction des circonstances. Dans un nouveau fichier Batch, copiez le texte suivant :

```
@echo off
IF {%1}=={c} set lecteur=c:\&goto Dir
IF {%1}=={d} set lecteur=d:\&goto Dir
IF {%1}=={e} set lecteur=e:\&goto Dir
@echo Saisissez le nom du fichier Batch suivi d'un espace puis appuyez sur la touche C D ou E
:Dir
DIR %lecteur% > %lecteur%\Sortie.txt
```

Nous nous servons du premier paramètre saisi par l'utilisateur à la suite du nom du script : %1. Si, par l'exemple, la touche appuyée est la lettre C alors nous définissons une variable nommée %Lecteur% qui sera égale à c:\ puis nous nous rendons à l'étiquette nommée :Dir. Si la touche choisie n'est ni C, D ou E alors un message sera affiché. Dans le cas contraire, un fichier nommé Sortie.txt sera créé à la racine du lecteur analysé.

Autoriser ou non les variables temporisées :

Créez un premier fichier Batch contenant ceci :

```
@echo off
SET variable=avant
IF "%variable%" == "avant" (
set variable=maintenant
IF "%variable%" == "maintenant" @echo Si vous voyez ceci le script marche
)
```

Si vous l'exécutez, le message ne sera pas affiché. Nous créons une variable nommée %avant%. Si notre variable correspond à la chaîne de caractères %avant% alors nous assignons à cette variable une nouvelle chaîne nommée %maintenant%. La dernière condition pose ceci : Si notre variable est égale à %maintenant% alors affiche un message. Comme la variable a été fixée sur %avant% la condition n'est pas remplie et donc le message pas affiché. Créez un autre fichier Batch avec cette fois-ci ce contenu :

```
@echo off
setlocal EnableDelayedExpansion
SET Variable=avant
IF "%Variable%" == "avant" (
SET Variable=après
IF "!Variable!" == "après" @echo Si vous voyez ceci le script marche
)
```

Dans ce dernier cas les variables sont assignées dynamiquement et leur affectation vient écraser celle qui leur avait été au préalable assignée. Il y a plusieurs manières d'assigner des variables de manière dynamique :

* Insérez une variable d'environnement en utilisant la commande "Setlocal EnableDelayedExpansion".
* Lancez l'interpréteur de commandes en utilisant le commutateur /V:ON.
* Modifiez directement le Registre :

1) Cliquez sur Démarrer/Exécuter puis saisissez : regedit
2) Ouvrez HKEY_CURRENT_USER\Software\Microsoft\Command Processor.

3) Dans le volet de droite créez une nouvelle valeur DWORD nommée DelayedExpansion.
4) Éditez cette valeur et affectez comme données de la valeur le chiffre 1.
5) Quittez puis relancez "cmd".

Il est possible d'opérer la même modification dans l'arborescence HKEY_LOCAL_MACHINE\Software\Microsoft\Command Processor si vous souhaitez que les modifications soient opérationnelles pour l'ensemble des utilisateurs de votre machine. Comparez ces deux fichiers Batch :

```
@echo off
setlocal EnableDelayedExpansion
SET Liste=
FOR %%a in (*) do SET Liste=%%a& ECHO !Liste!
```

```
@echo off
SET Liste=
FOR %%a in (*) do SET Liste=%%a& ECHO %Liste%
```

Dans ce dernier cas, rien ne sera affiché. Le premier script affichera le contenu du répertoire courant (*). Nous assignons à chaque nouvel objet trouvé (fichier ou répertoire) la variable %liste% que nous affichons en utilisant la commande "ECHO".

Déplacer la position des paramètres de commandes :

Dans un nouveau fichier de commandes appelé Test.bat, copiez le texte suivant :

```
@echo off
:Copie
shift
IF "%1"=="" GOTO Fin
COPY %1 c:\sauvegarde
GOTO Copie
:Fin
```

Saisissez par exemple : test *.txt *.doc *.bat *.bmp *.ppt *.eml

Nous allons donc copier tous les fichiers spécifiés du répertoire courant vers le répertoire C:\Sauvegarde. Le principe consiste à utiliser la commande "Shift" qui remplacera la variable %0 par la variable %1 (*.doc), la variable %1 par %2 (*.bat) une fois la première commande terminée (et ainsi de suite). En l'absence de la commande "Shift" nous chercherions désespérément un fichier introuvable... Il est donc possible de créer un fichier de commandes qui accepte plus de dix paramètres (de 0 à 9) puis que les arguments situés à partir de la dixième position seront décalés un à un afin d'occuper la variable %9. La commande spécifiant une condition nous permet d'éviter de rentrer dans une boucle sans fin. Si la variable actuelle est la même chose que rien ("") alors nous allons à l'étiquette :Fin. Dans un premier temps, nous avons donc cette commande : IF "*.txt"=="" GOTO fin. Comme *.txt est différent de rien, nous retournons à l'étiquette :Début. S'il n'y a plus aucun paramètre trouvé, nous poserons alors la condition suivante : IF ""=="" GOTO Fin. et là, dans ce cas, le script prend fin ! Il y a une manière légèrement différente qui nous permet de spécifier le répertoire de destination. Dans un nouveau fichier Batch copiez ces lignes :

```
@echo off
SET variable=%1
:Copie
shift
IF "%1"=="" GOTO Fin
COPY %1 %variable%
GOTO Copie
:Fin
```

Saisissez le nom de votre fichier Batch suivi du répertoire de destination et des fichiers à copier. Par exemple : test c:\sauvegarde *.txt *.xls *.doc *.bat *.bmp *.ppt *.eml

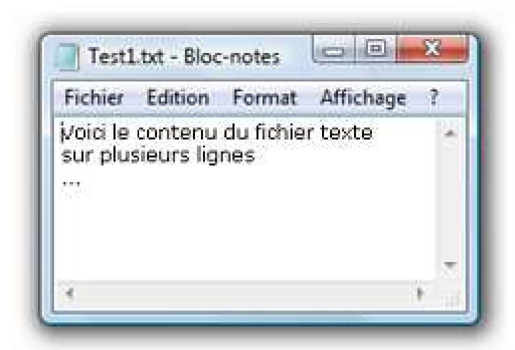
La première variable %1 passe le premier argument saisi : Le répertoire de destination. Nous définissons cette première variable sous le nom de %variable%. L'étiquette :Copie annonce les commandes permettant le processus de copie. La commande "Shift" décale la variable : %1 devient %0. Nous avons donc simplement défini le répertoire de destination. Pour le reste le principe de fonctionnement est strictement identique au script précédent.

Exécuter une commande pour un jeu de fichiers :

La syntaxe générale est celle-ci :

For {%variable|%%variable} in (jeu) do commande [Options_Ligne_Commande]

Prenons tout d'abord un exemple...



Si nous souhaitons afficher le contenu de tous les fichiers .txt et .doc du répertoire courant, nous saisirons :

```
@echo off
FOR %%1 IN (*.doc *.txt) DO type %%1
PAUSE
@echo on
```

C:\Windows\system32\cmd.exe

```
Voici le contenu du fichier texte
sur plusieurs lignes
...Appuyez sur une touche pour continuer...
```

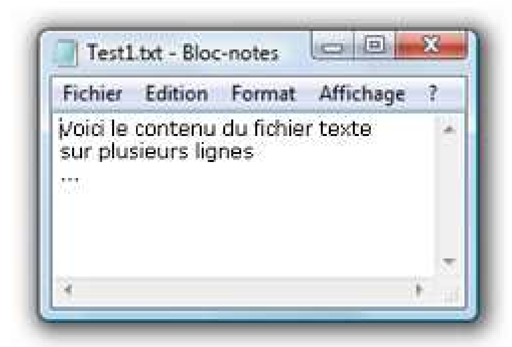
Signalons que la variable peut être un chiffre ou une lettre (par exemple, **%a** fait aussi bien l'affaire). Si nous utilisons cette commande dans un fichier de commande le signe % doit être redoublé. Jeu représente les noms de fichiers et doit être placé entre parenthèses. Les mots clés in et do sont possibles lors de l'utilisation de la commande "**FOR**". Nous pouvons les traduire par ceci : Pour tous les fichiers présents dans (**IN**) ce jeu de fichiers exécute (**DO**) cette commande. Il vous est possible d'utiliser les paramètres spécifiques à chaque commande (Options_Ligne_Commande). Une manière de lister les noms seuls des fichiers exécutables du répertoire courant est de saisir :

```
@echo off
FOR %%a IN (*.exe) DO @echo %%a
PAUSE
@echo on
```

Exécuter une commande pour un jeu de fichiers :

La syntaxe générale est celle-ci :

For {%variable|%%variable} in (jeu) do commande [Options_Ligne_Commande]



Prenons tout d'abord un exemple...

Si nous souhaitons afficher le contenu de tous les fichiers .txt et .doc du répertoire courant, nous saisirons :

```
@echo off
FOR %%1 IN (*.doc *.txt) DO type %%1
PAUSE
@echo on
```

```
C:\Windows\system32\cmd.exe
Voici le contenu du fichier texte
sur plusieurs lignes
...Appuyez sur une touche pour continuer...
```

Signalons que la variable peut être un chiffre ou une lettre (par exemple, **%a** fait aussi bien l'affaire). Si nous utilisons cette commande dans un fichier de commande le signe % doit être redoublé. Jeu représente les noms de fichiers et doit être placé entre parenthèses. Les mots clés in et do sont possibles lors de l'utilisation de la commande "**FOR**". Nous pouvons les traduire par ceci : pour tous les fichiers présents dans (**IN**) ce jeu de fichiers, exécute (**DO**) cette commande. Il vous est possible d'utiliser les paramètres spécifiques à chaque commande (Options_Ligne_Commande). Une manière de lister les noms seuls des fichiers exécutables du répertoire courant est de saisir :

```
@echo off
FOR %%a IN (*.exe) DO @echo %%a
PAUSE
@echo on
```



La simple liste de tous les fichiers (à l'exception des répertoires) s'obtient en saisissant :

```
@echo off
FOR %%a in (*) DO @echo %%a
PAUSE
@echo on
```

Pour avoir en plus les sous-répertoires :

```
@echo off
FOR /r %%a in (*) DO @echo %%a
PAUSE
@echo on
```

Exécuter une commande de manière récursive :

En Invite de commandes si vous souhaitez afficher tous les sous-répertoires appartenant à votre profil vous saisirez :

FOR /r "%userprofile%" %a IN (.) DO @echo %a

Quand Jeu est simplement un point, c'est l'arborescence principale qui est énumérée. Si vous souhaitez simplement lister les sous-répertoires de l'emplacement par défaut saisissez :

FOR /r %a in (.) DO @echo %a

Afin de supprimer tous les fichiers .doc de l'emplacement par défaut il suffit de saisir :

FOR /r %a in (*.doc) DO DEL /q "%a"

Gestion des noms comprenant des espaces :

La commande suivante ne fonctionnera pas si le nom d'un des fichiers .doc ou .txt contient des espaces :

FOR %a in (*.doc *.txt) DO COPY %a c:\temp\

Vous devez dans ce cas mettre la seconde variable entre guillemets :

FOR %a in (*.doc *.txt) DO COPY "%a" c:\temp\

Effectuer une substitution de variable :

Il est possible de changer les variables de sortie. Par exemple, si nous saisissons cette commande :

FOR /r %a IN (*.txt) DO @echo %~na

Seuls les fichiers .txt seront affichés. La liste ci-dessous récapitule les variables de substitution ainsi que les combinaisons permises. Rappelons que la variable %a peut-être remplacé par n'importe quelle autre lettre de l'alphabet ou chiffre.

%~a : Développe %a en supprimant les guillemets.
%~fa : Développe %a en un nom de chemin complet.
%~da : Développe %a en une lettre de lecteur seulement.
%~pa : Développe %a en un chemin seulement.
%~na : Développe %a en un nom de fichier seulement.
%~xa : Développe %a en une extension de fichier seulement.
%~sa : Développe le chemin afin qu'il ne contienne que des noms courts.
%~aa : Développe %a jusqu'aux attributs du fichier.
%~ta : Développe %a jusqu'à la date et l'heure du fichier.
%~za : Développe %a jusqu'à la taille du fichier.
%~$PATH:a : Recherche les répertoires énumérés dans la variable d'environnement et Développe %a jusqu'au nom complet du premier répertoire trouvé.
%~dpa : Développe %a en une lettre de lecteur et un chemin seulement.
%~nxa : Développe %a en nom de fichier et une extension seulement.
%~fsa : Développe %a en un nom de chemin complet avec des noms courts seulement.
%~dp$PATH:a : Recherche les répertoires énumérés dans la variable d'environnement pour %a et Développe jusqu'à la lettre du lecteur et au chemin du premier répertoire trouvé.
%~ftzaa : Développe %a en une ligne de sortie semblable à celle affichée par la commande "DIR".

Comparez les sorties affichées par ces trois commandes :

FOR /r %a IN (*.txt) DO @echo %~dpa
FOR /r %a IN (*.txt) DO @echo %~nxa
FOR /r %a IN (*.txt) DO @echo %~ftzaa

Signalons que la commande FOR /r %a IN (*.txt) DO @echo %~dp$PATH:a n'affichera que les arborescences contenant au moins un fichier Texte. Là aussi, comparez le texte de sortie avec cette commande :

(FOR /r %a IN (*.txt) DO @echo %~$PATH:a) > Sortie.txt

Nous avons simplement mis entre parenthèses la commande et redirigé son résultat dans un fichier .txt nommé Sortie.txt.

Décomposer une variable :

Dans un nouveau document Batch copiez ce contenu :

FOR /f "tokens=6-8 delims=/ " %%a IN ('echo. ^|date') DO SET jour=%%a&set mois=%%b& SET année=%%c

La commande "Date" affiche ce texte : "La date du jour est : 12/07/2004"

Nous devons donc récupérer le sixième bloc de caractères (12/07/2004) et, si nous enlevons les barres inversées, 8 caractères à partir de la fin du sixième bloc (12 07 2004). Trois valeurs nous intéressent : 12, 07 et 2004 que nous transformons en variables en utilisant la commande "**SET**". Pour vous en assurer saisissez cette commande : set jour

Nous pouvons faire la même opération pour la commande "Time" :

FOR /f "tokens=2-8 delims=:," %%a in ('echo. ^|time ^| find "actuelle"') do set HH=%%a&set HM=%%b& set HS=%%c& set HDS=%%d

Dans ce cas le principe est un peu différent : Si je saisis la commande Time, le message suivant apparaît : "L'heure actuelle est : 11:51:29,13" - "Entrez la nouvelle heure". Évidemment seule la première ligne nous intéresse... Nous allons demander que soit affichée la ligne qui contient le mot "actuelle" : echo. ^|time ^| find "actuelle"

Dans cette ligne nous allons extraire le second bloc puis les huit caractères en partant de la fin : tokens=2-8

Du résultat brut (11:51:29,13) nous allons effacer les signes : et ,(delims=:,). Quatre variables vont être définies : HH, HM, HS et HDS correspondant respectivement à l'heure, aux minutes, aux secondes et aux dixièmes de secondes. Nous devons donc spécifier quatre noms de variables : %a, %b, %c et %d. L'opérateur & permet d'exécuter plusieurs fois la commande "Set" en une seule ligne. Le tableau suivant récapitule les mots clés que vous pouvez utiliser :

Mot clé	Signification
eol=c	Indique un (et un seul) caractère de fin de ligne
skip=n	Spécifie le nombre de lignes à sauter à partir du début du fichier.
delims=xxx	Spécifie un séparateur.
tokens=x,y,m-n	Spécifie les jetons à passer à la commande For.
usebackq	Permet l'utilisation des guillemets.

Nous voyons plus en détail le principe d'utilisation d'un jeton au paragraphe suivant.

Principes d'utilisation d'un jeton :

Le mot clé "Tokens" sert à allouer un jeton pour chaque fichier analysé. Un jeton est sorte de permission temporaire accordée à un fichier afin que soit exécutée la commande spécifiée. Une fois cette dernière achevée, le jeton est alloué au prochain fichier trouvé et ainsi de suite. Si nous souhaitons activer l'attribut "Lecture seule" des fichier .doc nous pouvons saisir :

FOR /f "tokens=*" %a IN ('DIR /s /b *.doc') DO ATTRIB +r %a

Bien entendu, notre commande s'exécutera pour tous les fichiers .doc placés dans les répertoires et sous-répertoires de l'emplacement par défaut. Nous pouvons traduire cette commande par : "Pour chaque fichier trouvé quand je lance la commande '**DIR**' exécute celle qui suit le mot clé "**DO**". Il vous est également possible d'utiliser des itérations de plages de valeurs afin, par exemple, de créer rapidement dix répertoires nommés Test et numérotés de 1 à 10 :

FOR /l %a IN (1,1,10) DO @md test%a

Nous avons rajouté une arobase devant la commande "**Md**" afin de désactiver l'affichage de la sortie-écran. De la même façon, vous pouvez créer des "listes négatives" :

FOR /l %a IN (-10,1,10) DO @echo %a >> sortie.xls

Un document Excel sera créé avec une plage de valeurs comprises entre -10 et 10. La syntaxe est la suivante : Valeur_Début, Valeur_Incrémentation, Valeur_Fin. Créez un fichier texte nommé Fichier.txt contenant ceci :

Cellule1 Cellule2 Cellule3 etc.
-Cellule4 Cellule5 Cellule6 etc.
Cellule7, Cellule8, Cellule9 etc.
Cellule10 Cellule11 Cellule12 etc.

Saisissez cette commande si vous voulez afficher l'ensemble de Fichier.txt à l'exception de la seconde ligne :

FOR /f "eol=- tokens=1,2* delims=, " %a IN (fichier.txt) DO @echo %a %b %c

"eol=-" : Spécifie que les lignes commençant par un - seront ignorées.
"tokens=" : alloue un jeton puis un second pour chaque ligne explorée. Deux blocs de données seront adressés. L'astérisque permet d'allouer autant de variables que nécessaire pour recevoir le texte "restant". C'est plus simple d'utilisation que d'indiquer : "tokens=1,2,3,4" (puis que nous devons afficher quatre blocs de donnés).
"delims=, " : Nous permet de préciser que les jetons passés sont à chercher dans les blocs délimités par un espace ou une virgule.
"%a %b %c" : Permet de dénombrer le nombre de valeurs à afficher.

À titre de test saisissez cette commande :

FOR /f "eol=- tokens=1,2 delims=, " %a in (fichier.txt) DO @echo %a %b %c %d

Les deux dernières variables seront indiquées comme étant non "remplies" puisqu'un nombre insuffisant de jetons a été alloué. L'affichage des deux dernières colonnes s'obtient donc en saisissant :

FOR /f "eol=- tokens=3,4 delims=, " %a in (fichier.txt) DO @echo %a %b

Nous pouvons résumer la commande en disant : "Pour chaque ligne analysée, passe le troisième et le quatrième jeton jusqu'à ce que soient renseignées les variables %a et %b qui correspondront aux blocs de données n°3 et 4".

Récupérer différentes variables à partir d'un fichier :

Dans un nouveau document nommé Fichier.txt copiez ceci :

C:\Sauvegarde
C:\Test

Dans un nouveau fichier Batch copiez ceci :

```
@echo off
SET Fichier=%1
SET Dossier=%2
FOR /f %%a in (fichier.txt) DO CALL :Commande %%a
:Commande
SET source=%1
COPY %source%%fichier% %dossier%
```

Ce Batch se lance de cette façon : Nom_Batch *.txt C:\Temp. Nous récupérons les deux paramètres saisis par l'utilisateur (*.txt et C:\Temp) et définissant deux variables (Fichier et Dossier) qui correspondent respectivement à %1 et %2. Comme nous l'avons dit précédemment le signe % doit être redoublé quand il est utilisé dans un fichier Batch. À chaque ligne trouvée dans Fichier.txt nous appelons les commandes inclues dans l'étiquette :Commande. Le premier paramètre trouvé %1 (C:\Sauvegarde) est passé en tant que variable nommée %source%. La ligne suivante se contente d'effectuer le processus de copie en concaténant les variables de telle sorte que la commande s'exécute.

Analyser une chaîne de caractères :

Il vous est possible d'analyser le contenu d'un fichier d'une commande ou d'une variable d'environnement en les plaçant entre deux guillemets simples et inversés :

FOR /f "usebackq" %a IN ('SET') DO @echo %a

Le commutateur /f vous permet de spécifier plusieurs jetons d'analyse. Dans cet exemple, l'utilisation du mot clé Usebackq est nécessaire puisque Jeu est placé entre accents inversés. Si vous ne souhaitez n'avoir que les noms sans les chemins correspondants, saisissez :

FOR /f "usebackq delims==" %a IN ('SET') DO @echo %a

Le principe est de spécifier un jeu de séparateur. Dans notre exemple, la ligne "windir=C:\WINDOWS" sera coupée à partir du signe égal. Il ne s'affichera plus que "windir". Il vous tout à fait possible de faire la même opération sur un fichier nommé Fichier.txt :

FOR /f "usebackq delims=" %a in ('Fichier.txt') DO @echo %a

Le Bloc-notes s'ouvrira directement dans le fichier spécifié puisqu'il lui est demandé d'être exécuté... Curieusement, le mot-clé "usebackq" n'est jamais employé ! Comme nous le verrons dans les paragraphes suivants, il est en effet plus simple de saisir, par exemple, ces commandes :

FOR /f "delims==" %a IN ('SET') DO @echo %a
FOR /f "delims=" %a IN (Fichier.txt) DO @echo %a

Utiliser un filtre avec la commande FOR :

La commande "**FOR**" ne s'accommode pas d'un filtre de redirection. Dans l'exemple suivant le symbole de redirection | n'a aucun effet :

FOR %%a IN (*.txt) DO TYPE %%a | more

Le symbole | étant repris pour le compte de la commande "**FOR**" et non de "**TYPE**". Vous devez en passer par la création de deux scripts : L'un contiendra ceci :

FOR %%a IN (*.txt) DO test %%a

Nous pouvons aussi écrire :

FOR %%a IN (*.txt) DO CALL test %%a

Ou encore :

FOR %%a IN (*.txt) DO command /c test %%a

Dans l'autre fichier (appelé dans notre exemple Test) copiez ceci :

@type %1 | more

Les fichiers seront triés en fonction de leur nom.

Récupérer le résultat d'une commande en tant que variable :

Imaginons que nous souhaitons récupérer le nom du dernier fichier sans son extension et de nous en servir comme d'une variable, nous pouvons nous inspirer de ce script :

```
@echo off
FOR /f %%a IN ('DIR /b /od /a-d *.*') DO SET variable=%%~na
ECHO Le nom du dernier fichier est : %variable%
```

Le principe du script consiste à récupérer la dernière ligne affichée par la commande "**DIR**" et de transformer cette ligne en une variable. Pour ce faire, seul le nom du fichier seul est extrait (%~na).

Comparer si deux fichiers font la même taille :

Dans un nouveau fichier Batch copiez ce contenu :

```
@echo off
SET fichier1=%1
SET fichier2=%2
FOR %%a IN (%fichier1%) DO SET taille1=%%~za
FOR %%a IN (%fichier2%) DO SET taille2=%%~za
IF %taille1%==%taille2% (
ECHO Les tailles des fichiers sont identiques.
) ELSE (
ECHO Les tailles des fichiers ne sont pas identiques.
)
```

Deux variables récupèrent les deux paramètres saisis par l'utilisateur. Ce sont les deux noms de fichiers à comparer. Pour chaque fichier, nous procédons à l'extraction de la taille (%~za) et créons à chaque fois une variable. Si les deux tailles sont identiques, nous affichons un message de confirmation sinon nous signalons que les tailles trouvées ne sont pas identiques.

Créer un contrôle de saisie :

Dans le cas d'un fichier Batch nécessitant la saisie d'un paramètre vous pouvez insérer en début de script une routine de contrôle. Vous pouvez tester ces deux méthodes en saisissant ou non n'importe quelle variable en début de votre fichier batch :

IF [%1] ==[] ECHO Paramètres de commande manquants
IF [%1] equ [] ECHO Paramètres de commande manquants

Cela revient à dire : Si aucune chaîne de caractères n'a été saisie à la suite du nom du fichier de commandes affiche ce message : "Paramètres de commande manquants". Voici un autre exemple :

```
@echo off
IF {%1}=={} @echo Saisissez le nom du fichier&GOTO :eof
IF NOT EXIST %1 (@echo %1 n'existe pas) else @echo Le fichier existe !
```

La première ligne s'assure qu'à la variable %1 correspond une saisie quelconque. Si le paramètre qui suit le nom de la commande {%1} est égale == à rien {} ou, plus exactement, si rien n'a été saisi au clavier alors un message d'avertissement sera affiché. Si le paramètre saisi ne correspond (dans ce cas) à aucun fichier existant, alors affiche cet autre message d'avertissement. L'étiquette :GOTO:eof permet de quitter le fichier de commande si la condition posée est vraie. Si la condition est fausse, la commande suivante sera exécutée. Cela évite d'insérer une étiquette :fin à la fin de notre script. Il y a une méthode plus simple : inspirez-vous de ce script :

```
@echo off
IF NOT %1[==[ IF EXIST %1 GOTO message
ECHO Le fichier n'existe pas ou rien n'a pas été précisé & GOTO:eof
: message
ECHO Bonjour !
```

De cette façon, nous vérifions en une seule commande que l'utilisateur a bien saisi un paramètre et que le fichier spécifié est bien présent.

Choisir à quel étage l'ascenseur va s'arrêter :

Dans un nouveau Batch copiez ce contenu :

```
@echo off
IF "%1" == "3" GOTO niveau3
IF "%1" == "2" GOTO niveau2
IF "%1" == "1" GOTO niveau1
IF "%1" == "" GOTO :eof
:niveau3
CD ..\..\..\
GOTO :eof
:niveau2
CD ..\..\
GOTO :eof
:niveau1
CD ..\
GOTO :eof
```

Copiez ce script dans un des chemins définis par la variable d'environnement %Path%. Saisissez à partir de n'importe quelle arborescence le nom du Batch suivi du numéro de niveau auquel vous souhaitez remonter. Si, par exemple, nous saisissons le chiffre 3 nous exécuterons cette commande : CD ..\..\..\.

7 – **Exemple de fichiers Batch** :

- Créer une page Web :

```
@echo off
SET site=C:\temp\index.htm
REM Ecrire la page index.htm
ECHO "<html><head><title>Présentation d'un site web</title></head><!-- " > %site%
ECHO " --><body><hl>Bienvenue sur ma page</hl><!-- " >> %site%
ECHO " --><div style=color:red;text-align:center;>Je m'appelle..., <br/> et je veux être...</div><!-- " >> %site%
ECHO " --><a href=http://www.weboplanet.com>consulter les cours</a></body></html>" >> %site%
REM Afficher le contenu du fichier puis le modifier
TYPE %site%
Notepad %site%
PAUSE
REM Lancement d'Internet Explorer
"C:\program files\internet explorer\iexplore.exe" "%site%"
@echo on
```

- Auto-supprimer le fichier Bacth :

```
@echo off
pushd %1
del /q *.bat
for /f "Tokens=*" %%G in ('dir /B') do rd /s /q "%%G"
popd
@echo on
```

- Trouver la taille d'un ou plusieurs fichiers : (Exemple avec l'extension *.mp3)

```
@echo off
for %%i in (*.mp3) do (
echo %%~ni pese %%~zi Octet.
)
pause
```

```
@echo on
```

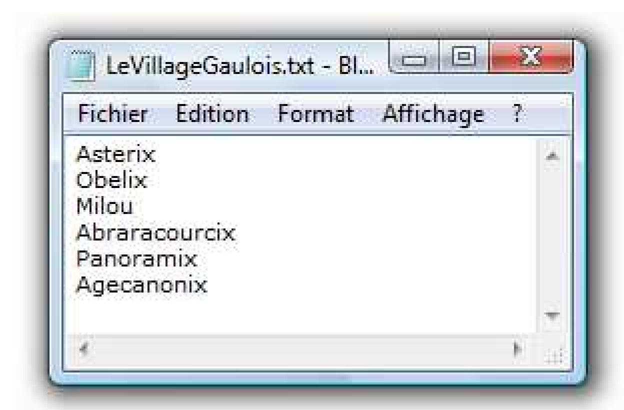
```
C:\Windows\system32\cmd.exe
3T - Anything For You pese 5144485 Octet.
3T - I Need You pese 3788068 Octet.
3T - Waiting For Love pese 3633243 Octet.
Appuyez sur une touche pour continuer...
```

- Pour bloquer l'accès au contenu d'un répertoire par un mot de passe sans cryptage et logiciel : Créer un Nouveau Dossier qu'on nommera « 001 » par exemple et y mettre votre fichier **.bat** suivant à l'intérieur.

```
@echo off
title Folder Private
if EXIST "Control Panel.{21EC2020-3AEA-1069-A2DD-08002B30309D}" goto UNLOCK
if NOT EXIST Private goto MDLOCKER
:CONFIRM
echo Etes vous sure de vouloir bloquer les fichiers Y=Oui N=Non(Y/N)
set/p "cho=>"
if %cho%==Y goto LOCK
if %cho%==y goto LOCK
if %cho%==n goto END
if %cho%==N goto END
echo Invalid choice.
goto CONFIRM
:LOCK
ren Private "Control Panel.{21EC2020-3AEA-1069-A2DD-08002B30309D}"
attrib +h +s "Control Panel.{21EC2020-3AEA-1069-A2DD-08002B30309D}"
echo Folder locked
goto End
:UNLOCK
echo Enter password to unlock folder
set/p "pass=>"
if NOT %pass%== tonpassici goto FAIL
attrib -h -s "Control Panel.{21EC2020-3AEA-1069-A2DD-08002B30309D}"
ren "Control Panel.{21EC2020-3AEA-1069-A2DD-08002B30309D}" Private
echo Bien Bloquer GG
goto End
:FAIL
echo Invalid password
goto end
:MDLOCKER
md Private
echo Private created successfully
goto End
:End
```

Là où il y est écrit "tonpassici", mettre à la place votre mot de passe (ex : if NOT %pass%== Bonbon goto FAIL). Exécuter votre fichier .bat à l'intérieur du dossier « 001 », une fenêtre de commande dos s'ouvre, taper votre mot de passe en respectant la casse ("tonpassici" dans notre exemple) puis OK (appuyer la touche entrée). Un nouveau dossier s'ouvre automatiquement et porte le nom "Private". Ouvrir ce dossier (Private) et y mettre ce que vous voulez. Une fois que vous avez fini ceci, ré exécuter encore votre fichier .bat (Confidentiel01.bat), une fenêtre de commande dos s'ouvre, taper **Y** pour bloquer les fichiers, le répertoire "Private" n'apparaît plus. Pour débloquer, recliquer 2 fois sur le fichier .bat et entrer votre mot de passe. Il ne vous reste ensuite plus qu'à crypté votre fichier .bat à l'aide d'un logiciel spécialisé. Mais le système n'est tout de même pas sécurisé, car en décochant la case « Masquer les fichiers protégés par le système » dans les propriétés de Windows, le fichier « Private » apparaîtra.

- Remplacer une chaîne de caractères dans un fichier texte définit : Dans l'exemple suivant, on remplace "Milou" par "Idefix" dans le fichier texte "LeVillageGaulois.txt" sous "C:\Temp".




```
@echo off
REM Remplacer une chaîne de caractères dans un fichier texte définit
REM On définit le chemin :
SET chemin=C:\Temp\
SET motrechercher="milou"
SET motremplacerpar=Idefix

REM Si le fichier n'existe pas ou n'a pu été trouvé, on arrête tout
IF not exist %chemin%LeVillageGaulois.txt goto fin

REM On cherche le numéro de ligne et on l'écrit dans un fichier tmp.txt
REM Cette valeur est sous la forme d'une chaîne : [numeros de ligne]valeur recherchee
TYPE %chemin%LeVillageGaulois.txt | FIND /n /i %motrechercher% > %chemin%tmp.txt

REM On récupère cette valeur pour la mettre dans une variable
FOR /f %%i IN (%chemin%tmp.txt) DO SET numligne=%%i

REM On en profite pour vérifier que quelque chose a bien été trouvé
IF "%numligne%" == "" goto rientrouvee

REM On récupère le numéro de ligne dans la chaîne: [numeros de ligne]valeur recherchee
SET numligne=%numligne:~1,1%
SET /a numligne=numligne-1

REM Création d'un fichier temporaire à partir du fichier source,  mais sans la ligne trouvée
TYPE %chemin%LeVillageGaulois.txt | FIND /v /i "milou" > %chemin%tmp.txt

REM On efface le fichier source.
DEL %chemin%LeVillageGaulois.txt

REM On boucle sur toutes les lignes du fichier temporaire en passant chaque ligne en paramètre à la subroutine
(%%i)
SET compteur = 1
FOR /f %%i IN (%chemin%tmp.txt) DO CALL :subroutine %%i

REM La boucle est terminée on efface le fichier temporaire
DEL %chemin%tmp.txt
ECHO Modification terminee
PAUSE
goto fin

:rientrouvee
DEL %chemin%tmp.txt
ECHO La ligne recherchee n'a pu etre trouvee
PAUSE
goto fin

:subroutine
REM On recree le fichier source avec chaque ligne du fichier temporaire via le parametre %1
ECHO %1 >> %chemin%LeVillageGaulois.txt
SET /a compteur=compteur+1
REM Si compteur est = au numero de ligne qui précède la ligne à remplacer on ajoute la nouvelle ligne
IF %compteur% == %numligne% ECHO %motremplacerpar% >> %chemin%LeVillageGaulois.txt

:fin
```

- Copier des fichiers de différentes extensions :

```
@echo off
FOR %%a in ( %1 %2 %3 %4 %5 %6 %7 %8 %9) do xcopy *.%%a c:\sauvegarde /s /y
@echo on
```

Lancez votre fichier Batch en le faisant suivre du nom des extensions voulues. Nous créons une boucle de contrôle qui va pour chaque paramètre passé à la suite de notre fichier Batch entamer le processus de copie vers le répertoire C:\Sauvegarde.

- Spécifier un groupe de commandes dans un fichier :

Créez un fichier nommé Fichier.txt dans lequel vous allez copier ce contenu :

attrib -h -s
attrib -a
attrib -r
attrib

Dans un nouveau fichier Batch nommé Test.bat copiez ce contenu :

```
@echo off
FOR /f "delims=" %%a in (fichier.txt) do %%a *.%1
@echo on
```

La commande permettant d'exécuter ces quatre opérations sur tous les fichiers *.bmp est celle-ci : test bmp. Le principe est de lire le contenu du fichier ligne par ligne et pour chaque objet trouvé par la commande "FOR". La variable %1 remplace le premier paramètre saisi par l'utilisateur à la suite du nom du fichier Batch. Le mot-clé "Delims" est nécessaire sinon il n'est pas tenu compte des espaces dans le parcours des commandes inscrites dans le fichier.

- Créer un fichier texte en se servant de la commande Echo :

Nous nous sommes inspirés d'une suggestion faite par Jerold Schulman à partir de son site : www.jsiinc.com. Le principe est très simple : nous allons encore une fois nous servir des parenthèses. Imaginons que nous souhaitions désinstaller complètement la machine virtuelle Java, il nous suffira de supprimer certaines données de l'Explorateur Windows ainsi que du Registre Windows. Dans ce dernier cas nous allons créer un fichier .reg qui une fois enregistrée permettra d'opérer les modifications nécessaires. Créez un nouveau fichier Batch en saisissant ceci :

```
@echo off
rd /s /q %SystemRoot%\JAVA
del /q %systemroot%\inf\java.pnf
del /q %systemroot%\system32\jview.exe
del /q %systemroot%\system32\wjview.exe
(
@echo Windows Registry Editor Version 5.00
@echo/
@echo [-HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Java VM]
@echo/
@echo [-HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Internet Explorer\AdvancedOptions\JAVA_VM]
@echo/
) >"%Temp%\Java.reg"
regedit /s "%Temp%\Java.reg"
del /q "%Temp%\Java.reg"
@echo on
```

Dans un premier temps, nous supprimons le répertoire Java placé dans \Windows. À partir de ce même emplacement, nous effaçons les fichiers Java.pnf, Jview.exe et Wjview.exe. Il nous reste plus qu'à supprimer ces deux clés du Registre :

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Java VM
et
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Internet Explorer\AdvancedOptions\JAVA_VM

Il suffit dans ce cas d'ajouter le préfixe - au début du nom de la clé. Pour ce faire nous allons utiliser la commande Echo en créant un fichier de commande temporaire placée entre deux parenthèses. De cette façon, c'est l'ensemble des lignes commençant par @echo qui seront redirigées vers un seul et même fichier .reg. Il nous reste plus qu'à l'enregistrer dans le répertoire Temp sous le nom de Java.reg. La commande regedit /s nous permet ensuite de fusionner les informations contenues dans ce fichier au Registre Windows. Pour terminer, il nous reste plus qu'à supprimer le fichier Java.reg.

- Nommer un fichier de manière incrémentielle :

Dans un nouveau fichier Batch copiez ce contenu:

```
@echo off
set nombre=
:Boucle
if not exist Document%nombre%.txt goto Copie
set /a nombre+=1
goto Boucle
:Copie
copy Document.txt Document%nombre%.txt
@echo on
```

Nous nous proposons dans ce Batch de renommer automatiquement un fichier nommé Document.txt en le numérotant à chaque fois. Nous créons une structure de boucle avec un "compteur" qui n'est pas encore initialisé. Nous assignons au compteur un nom de variable nommée %nombre%. Le script démarre ensuite les commandes contenues dans l'étiquette :Boucle. Si un fichier portant ce nom existe déjà nous incrémentons la variable %nombre% d'un pas de 1 et nous retournons au début de notre boucle. Sinon nous exécutons les commandes spécifiées sous l'étiquette :Copie : le fichier précédent est copié puis renommé en ajoutant à son nom la variable %nombre%.

- Copier les fichiers en les renommant par incrémentation :

C'est un prolongement du script précédent... Dans un nouveau fichier Batch copiez ce contenu :

```
@echo off
set compteur=0
set rep1= c:\source
set rep2= c:\destination
for /f "tokens=1-2 delims=." %%a in ('dir /b %Rep1%') do (
call Copie "%%a" "%%b"
)
@echo on
```

Dans un nouveau fichier Batch (dans notre exemple, nommé Copie.bat) copiez ce contenu :

```
@echo off
set nom=%~1
set extension=%~2
set /a compteur+=1
set fichier=%nom%%compteur%.%extension%
copy %rep1%\"%nom%.%extension%" %rep2%\"%fichier%"
@echo on
```

Le script va renommer automatiquement chaque fichier copié en leur attribuant un numéro de traitement. Nous définissons une variable et deux répertoires. La troisième ligne permet de le supprimer entre le nom du fichier et son extension ("delims=.") et de décomposer le résultat en deux blocs distincts ("tokens=1-2"). Pour chaque objet trouvé dans le répertoire "source" nous appelons l'autre fichier Batch. Au nom et à l'extension du fichier sont attribuées respectivement deux variables nommées %nom% et %extension%. La variable %compteur% est incrémenté d'un pas de 1. Une variable nommée %fichier% est créée en ajoutant à la suite de la variable %nom% la valeur de %compteur%. Nous copions le fichier correctement renommé dans le répertoire de destination. Les guillemets permettent de copier également les noms de fichiers comportant des espaces. Une autre méthode permettant de renommer des fichiers. Nous nous inspirons d'une solution proposée par Timo Salmi à partir de cette page Web : www.uwasa.fi/~ts/http/http2.html#batch. Dans un nouveau fichier Batch copiez ce contenu :

```
@echo off
 set compteur=1
 for /f %%a in ('dir c:\scripts\*.txt /b /s /-c /a:-d-s-h') do (
 call :menu %%a)
 set compteur=
 goto :eof
 :menu
 set fichier=%~d1& set chemin=%~p1& set nom=%~n1& set extension=%~x1
 ren %fichier%%chemin%%nom%%extension% %nom%.%compteur%%extension%
 set /a compteur+=1
goto :eof
@echo on
```

Nous initialisons la variable %compteur% sur 1. Nous effectuons un balayage de la sortie de la commande "Dir" en appelant pour chaque ligne trouvée les commandes placées sous l'étiquette :menu. Chaque fichier sera décomposé entre son nom et son extension. Puis il sera renommé en accolant la valeur de la variable %compteur%. Cette dernière sera à chaque fois incrémentée d'un pas de un.

- Supprimer tous les fichiers dans tous les répertoires spécifiés :

Dans cet exemple nous allons supprimer tous les fichiers contenus dans tous les répertoires nommés Temp.

```
@echo off
for /f "delims=" %%a in ('dir c:\ /b /ad /s ^| findstr /i \%1') do echo del "%%a\*.*"
@echo on
```

Saisissez le nom du fichier Batch suivi du nom du répertoire à "vider". La commande findstr /i \Temp nous permet de retrouver toutes les arborescences contenant la chaîne de caractères "Temp" et qui seront affichées par "Dir". Vous devez préfixer le symbole de redirection | du signe ^ puisqu'il fait partie des caractères réservés. Enfin, nous allons supprimer tous les fichiers présents dans chacune des arborescences trouvées.

- Une manière de supprimer des fichiers différents :

Dans un nouveau fichier Batch (nommé Test) copiez ce contenu :

```
@echo off
:Boucle
del *.%1
shift
if not "%1" == "" goto Boucle
@echo on
```

Il vous suffire alors de saisir ce type de commande : test txt doc exe. Le script démarre en supprimant les fichiers portant l'extension du premier paramètre saisi par l'utilisateur à la suite du nom du fichier Batch. La commande "Shift" permet de décaler la variable %1 ce qui permet de traiter le second paramètre (doc). Si une nouvelle variable est trouvée (si %1 n'est pas égale à "") alors nous reprenons le script à partir de l'étiquette :Boucle.

- Une manière de déplacer des fichiers différents :

Dans un nouveau fichier Batch (nommé Test) copiez ce contenu :

```
@echo off
for %%a in (%1 %2 %3 %4 %5 %6 %7 %8 %9) do move "%%a" c:\destination
@echo on
```

Il vous suffire alors de saisir ce type de commande : test *.txt *.doc *.exe. Là encore, le principe est très similaire à ce qui était expliqué auparavant sauf sue nous n'opérons pas de décalage de variable en nous servant de la commande "Shift". Cela revient à saisir directement cette commande :

```
@echo off
for %a in (*.txt *.doc *.exe) do move "%a" c:\destination
@echo on
```

Il est préférable de placer entre guillemets la seconde variable %a si les noms des fichiers traités comportent des espaces.

- Ajouter ou remplacer un ou plusieurs fichiers :

Dans un nouveau fichier Batch (nommé test) copiez ce contenu :

```
@echo off
replace %1\*.* %2 /a
replace %1\*.* %2 /u
@echo on
```

Saisissez, par exemple, cette commande : batch c:\source c:\destination. Opérez des modifications dans les fichiers placés dans le répertoire source puis à nouveau lancez la même commande en appuyant sur la touche F3. Il vous est possible de synchroniser les deux répertoires en remplaçant le contenu du fichier Batch par ceci :

```
@echo off
replace %1\*.* %2 /a
replace %1\*.* %2 /u
replace %2\*.* %1 /a
replace %2\*.* %1 /u
@echo on
```

- Décomposer la date d'aujourd'hui :

Voici une méthode assez simple. Dans un nouveau fichier Batch copiez ce contenu :

```
@echo off
for /f "tokens=1-3 delims=/" %%f in ("%date%") do (
set datation=%%f%%g%%h
set jour=%%f
set mois=%%g
set année=%%h)
echo %datation%
echo J=%jour% M=%mois% A=%année%
@echo on
```

Dans ce cas, nous procédons à l'analyse de la variable %date% tel qu'elle apparaît en saisissant ceci : echo %date%. En nous servant du mot clé "delims=/" nous supprimons les barres tranversales. Le mot-clé "tokens" nous permet de passer les trois blocs de données en trois variables nommées %f, %g et %h. Le reste du script n'est qu'un jeu sur les variables : Nous considérons qu'il est plus rapide de se servir d'une variable composée d'une seule lettre que d'un mot. Une autre solution consiste à extraire directement les informations utiles à partir de la variable prise textuellement :

```
@echo off
set jour=%date:~0,2%
set mois=%date:~3,2%
set année=%date:~5,4%
@echo Nous sommes le %jour%/%mois%/%année%
@echo on
```

- Insérer une date dans un nom de fichier :

Ceci peut en être un exemple :

```
@echo off
echo sauvegarde_%date:~0,2%%date:~3,2%%date:~6,4%.txt
@echo on
```

Nous pouvons imaginer toute sorte de "formatage" de la variable %date%...

- Lister les fichiers correspondant à une date de modification :

Nous inspirons d'un exemple de script proposé par Timo Salmi à partir de son site web : www.uwasa.fi/~ts/http/http2.html#batch. Dans un nouveau fichier Batch copiez ce contenu :

```
@echo off
set dossier=%1
set année=%2
set dossier1=dir /b /s %dossier%
set année1=findstr /l /c:"%année%"
for /f "tokens=*" %%c in ('%dossier1%') do (
set fichier=%%c
call :commande
)
goto :eof
:commande
for /f %%c in ('dir "%fichier%" ^|%année1%') do @echo %fichier%
@echo on
```

La syntaxe permettant de lancer ce Batch est celle-ci : Nom_Batch Répertoire Année. Par exemple : test c:\source 2004. Nous définissons quatre variables :

%1 : le premier paramètre saisi par l'utilisateur qui sera le nom du répertoire.
%2 : le second paramètre saisi par l'utilisateur qui sera l'année de modification.
%dossier1% : le résultat de la commande "Dir" qui trie les fichiers en fonction de leur date de modification.
%année1% : le résultat d'une recherche portant sur l'expression "2004" si l'année spécifiée est 2004.

Pour chaque ligne trouvée dans la sortie de la commande "Dir" il est alors demandé de :

-> Définir une variable nommée %c
-> D'appeler les instructions spécifiées sous l'étiquette :commande.

Chaque ligne contenant l'expression définie par la variable %année% sera affichée en utilisant la commande "Echo".

- Lister tous les fichiers modifiés aujourd'hui :

Saisissez cette commande :

```
@echo off
dir *.* /o:n | find "%date%" | more
@echo on
```

- Créer un dossier portant le nom de la date d'aujourd'hui :

Dans un nouveau fichier Batch copiez ce contenu :

```
@echo off
for /f "tokens=1-3 delims=/" %%a in ('date /t') do set dossier=c:\sauvegarde\%%a%%b%%c
md %dossier%
copy c:\test\*.txt %dossier%
@echo on
```

Nous nous proposons dans cet exemple de copier tous les fichiers Texte placés dans C:\Test dans un répertoire que nous allons créer et qui portera le nom de la date d'aujourd'hui. Le principe est assez simple :

Nous nous servons de la commande "For" pour extraire trois variables "tokens=1-3" de la commande "date /t". Une fois ces trois variables extraites nous créons une variable nommée %dossier% et qui correspond à un répertoire virtuel placé dans C:\Sauvegarde. Le nom de ce répertoire est la date du jour. Afin d'éviter un message d'erreur, nous nous servons du paramètre "delims=/" afin de soustraire du nom les barres transversales. Nous créons (Md) un répertoire portant le nom de la variable créée (%Dossier%). Pour mieux comprendre le mécanisme de ce script, saisissez ces commandes :

```
for /f "tokens=1-3 delims=/" %a in ('date /t') do set Dossier=c:\Sauvegarde\%a%b%c
```

À la suite du prompt sera affichée la date du jour...

```
md %dossier%
```

Vous pouvez tester ce script an accédant aux propriétés de "Date et heure" et en changeant momentanément la date du jour. Le même script peut être utilisé pour réaliser des sauvegardes des fichiers en les rangeant pas heure. Il suffit de modifier la première ligne en inscrivant :

```
for /f "tokens=1-2 delims=:" %%a in ('time /t') do set Dossier=C:\Sauvegarde\%%a%%b
md %dossier%
```

- Copier des fichiers en fonction de la date d'aujourd'hui :

Servez-vous de ce modèle de script :

```
@echo off
for /f "tokens=1-3 delims=/ " %%a in ('date/t') do set MMDDYY=%%b-%%a-%%c
xcopy c:\source\*.* c:\destination /D:%MMDDYY%
@echo on
```

Il y a un piège : la commande "Xcopy" exige avec le commutateur /d que le mois soit spécifié avant l'indication du jour. Pour le reste, nous extrayons de la commande "Date /t" trois variables (mois, jour, année) qui nous servira à spécifier à partir de quelle date de dernière modification les fichiers seront inclus dans le processus de copie.

- Copier différentes sauvegardes d'un même fichier en les datant :

Nous nous inspirons dans cet exemple d'une suggestion faite par Jerold Schulman à partir de son site Web : www.jsiinc.com. Dans un nouveau fichier Batch copiez ce contenu :

```
@echo off
setlocal
for /f %%a in ('dir /b/x c:\source') do call :début %%a
endlocal
goto:eof
:début
set Nom1=%1
for /f "tokens=1,2,3 delims= /" %%a in ('date /t') do set Date1=%%a%%b%%c
for /f "tokens=1,2 delims=: " %%d in ('time/t') do set Heure1=%%d%%e
for %%f in (%Nom1%) do (
set Nom2=%%~nf
set Extension=%%~xf
)
set Nom3=%Nom2%-%Date1%-%Heure1%%Extension%
copy c:\source\%Nom1% c:\destination\%Nom3%
@echo on
```

Dans ce cas, le principe est de créer des sauvegardes des fichiers placés dans le répertoire Source dont les différentes versions seront datées puis classées dans un répertoire nommé Destination. Signalons tout de suite que ce script ne fonctionne pas si le fichier contient des espaces ou des caractères réservés. Les lignes 1 et 3 nous permettent de réinitialiser à chaque exécution du script les variables %nom1%, %nom2% et %extension%. Une fois cette précaution prise, la commande goto:eof nous permet de sortir du fichier de script. Le script proprement démarre en ligne n°2. Pour chaque ligne affichée par la commande "Dir" nous allons exécutez les commandes placées sous l'étiquette :début. Une variable nommée %Nom1% nous permet de récupérer le nom du fichier affiché par "Dir". Nous décomposons le nom complet du fichier en son nom (%%~nf) puis son extension (%%~xf). Par ailleurs, nous procédons à l'extraction des commandes "Date /t" et "Time /t" des valeurs qu'elles contiennent. Enfin, nous créons une nouvelle variable nommée %Nom3% qui est la concaténation des variables obtenues précédemment. Ne reste plus qu'à enclencher le processus de copie puis recommencer le même processus pour le nom de fichier suivant.

- Changer la date de modification d'un fichier sans le modifier :

Ce type de commande le fait :

```
@echo off
copy /b /y "fichier.txt" +
@echo on
```

Cela revient à ajouter au fichier que nous copions un fichier inexistant. Aucune modification n'est alors apportée au fichier original. En l'absence du signe + il serait indiqué qu'il n'est pas possible de copier un fichier sur lui-même.

- Créer des scripts pour la gestion des répertoires :

Nous allons écrire un script qui cherche si un répertoire portant le nom indiqué existe bien sur un des lecteurs spécifiés. Dans un nouveau fichier Batch saisissez ce contenu :

```
@echo off
set Liste=C,D,E et F
for %%a in (%Liste%) do for /f "delims=" %%b in ('dir /s /b /ad "%%a:\" 2^>NUL ^| findstr /i "\%~1$"') do (
set dossier=%%b
goto :Sortie
)
echo %1 introuvable dans %Liste%
goto :eof
:Sortie
echo %dossier%
@echo on
```

Il suffit de saisir le nom du script suivi du nom du répertoire recherché. Nous définissons une variable nommée %liste% qui contient les noms des lecteurs. Aussi étonnant que cela puisse paraître la syntaxe n'a strictement aucune importance. Seul compte l'indication des lettres de lecteur. Suivent deux commandes "For" imbriquées l'une dans l'autre. Afin de mieux comprendre le fonctionnement de la seconde, saisissez ceci en Invite de commandes :

dir /s /b /ad "c:\" 2>nul | findstr /i "\destination"

Nous recherchons dans la sortie de la commande "Dir" une occurrence qui sera "\destination". En toute logique, si un répertoire nommé Destination est trouvé il sera repéré par la barre transversale qui le précède. La redirection 2>nul permet d'éviter l'affichage d'éventuels messages d'erreur. Rappelons que l'emploi de l'accent circonflexe est nécessaire pour préfixer les symboles de redirection > et | à partir du moment qu'ils sont utilisés dans un fichier de script. La variable %\~1$ permet de récupérer le chemin absolu qui mène au répertoire. La première ligne trouvée (%b) est passée en tant que variable et elle sera nommée %dossier%. Il ne reste plus qu'à se rendre à l'étiquette :Sortie et à afficher la variable obtenue.

- Récupérer l'emplacement par défaut :

Afin de récupérer le chemin du répertoire courant, inspirez-vous de ce Batch :

```
@echo off
 for /f "tokens=*" %%a in ('cd') do set chemin=%%a
echo %chemin%
@echo on
```

Nous nous servons simplement de la commande "Cd" pour extraire le chemin du prompt. Oui, cela peut paraître quelque peu curieux, mais ça fonctionne !

- Supprimer un nombre déterminé de fichiers dans un répertoire donné :

Dans un nouveau fichier Batch nommé Test copiez ce contenu :

```
@echo off
```

```
set répertoire=%1%
set fichier=%2%
set nombre=%3%
for /f "skip=%nombre%" %%a in ('dir /a:-d /b "%répertoire%\%fichier%"') do del %répertoire%\%%a
@echo on
```

En admettant que vous souhaitiez "écrémer" les 5 premiers fichiers .txt dans un répertoire Source, saisissez cette commande : test c:\source *.txt 5. Nous définissons trois variables correspondant aux trois paramètres saisis par l'utilisateur. Dans la sortie-écran affichée par la commande "Dir" nous "squizzons" les cinq premières lignes affichées correspondant aux cinq fichiers les plus anciens en bous servant du mot clé "skip=%nombre%". Il vous est aussi possible d'effectuer l'opération inverse en changeant l'ordre de tri :

dir /a:-d /o:-d /b

La variable %a sera chaque nom de fichier retenu que nous reprenons pour les supprimer un à un.

- Compter le nombre d'objets présents dans un répertoire :

Dans un nouveau fichier Batch, copiez ce contenu :

```
@echo off
set répertoire=%1
set /a compteurA=0
set /a compteurB=0
for /f %%a in ('dir /s /b /a:-d %répertoire%\*.*') do set /a CompteurA+=1
for /f %%a in ('dir /s /b /a:d %répertoire%\*.*') do set /a CompteurB+=1
echo %CompteurA% fichiers %compteurB% dossiers
@echo on
```

Saisissez le nom du script suivi du nom du répertoire à explorer. Nous récupérons le paramètre saisi par l'utilisateur afin de créer une variable nommée %répertoire%. Nous définissons deux compteurs qui sont chacun initialisés sur 0. La première commande va compter les fichiers. Nous nous servons du commutateur /a:-d dans la commande "Dir" afin de ne pas afficher les répertoires trouvés. La seconde commande utilise la fonctionnalité inverse de la commande "Dir".

- Supprimer le contenu d'un répertoire cible :

Dans un nouveau fichier batch copiez ce contenu :

```
@echo off
pushd %1
del /q "*.*"
for /f "tokens=*" %%a in ('dir /b /ad') do rd /s /q "%%a"
popd
@echo on
```

Saisissez le nom du fichier Batch suivi de l'emplacement et du nom du répertoire à "vider". Le répertoire "parent" restera intact tandis que l'intégralité de son contenu (fichiers et sous-répertoires) sera supprimée. Le principe est de récupérer l'arborescence spécifiée par l'utilisateur pour "pousser" (pushd) l'indicatif à l'emplacement voulu. La commande "Del" supprime tous les fichiers du nouveau répertoire par défaut. Nous utilisons la commande "Dir" afin de n'afficher que les répertoires. Pour chaque occurrence trouvée, nous nous servons de la commande "Rd". La dernière ligne nous permet de revenir à notre point de départ (popd).

- Analyser les fichiers :

Créez un nouveau fichier Batch contenant :

```
@echo off
set nom=%1
for /f %%a in ('find /v /c "" ^< %nom%') do set /a lignes=%%a
@echo Il y a %lignes% lignes dans le fichier %nom%
@echo on
```

Saisissez le nom du fichier batch suivi du nom du fichier à analyser. La seconde ligne du script nous permet de récupérer le nom du fichier en nous servant de la variable %1 qui désigne le premier paramètre saisi par l'utilisateur à la suite de la commande. La commande find /v /c "" affiche toutes les lignes qui contiennent autre chose que rien (""), opère un décompte puis affiche le total (/c). En admettant que la recherche porte sur un fichier nommé Fichier.txt, vous auriez pu saisir directement cette commande :

for /f %a in ('find /v /c "" ^< fichier.txt') do @echo Il y a %a lignes.

- Créer un fichier contenant un texte plusieurs fois répété :

Vous pouvez vous inspirer de cet exemple de fichier Batch :

```
@echo off
for %%a in (1 2 3 4 5 6 7 8 9) do echo Ligne %%a>>test.txt
@echo on
```

- Compter les lignes d'un fichier et en afficher un nombre déterminé :

Nous nous inspirons dans ce paragraphe et les autres scripts portant sur l'analyse des fichiers de certains exemples publiés par Ritchie Lawrence à partir de son site Web : www.commandline.co.uk/index.html. Dans un nouveau document Batch copiez les lignes suivantes :

```
@echo off
set fichier=%1
set nombre=%2
for /f %%a in ('find /v /c "" ^< %fichier%') do set /a lignes=%%a
@echo Il y a %lignes% lignes dans le fichier %fichier%.
if %nombre% geq %lignes% set /a A=0&goto Sortie
set /a A=%lignes% - %nombre%
:Sortie
more /e +%A% %fichier%
@echo on
```

Saisissez à la suite de votre "Batch" le nom du fichier suivi du nombre de lignes à afficher. Nous reprenons le principe du précédent Batch pour la première partie du script. Si le nombre saisi par l'utilisateur est supérieur ou égal au nombre de lignes trouvées nous exécutons la commande spécifiée sous l'étiquette :Sortie. Dans le cas contraire, nous définissons la variable %A comme étant le produit de la soustraction entre le nombre de lignes et le nombre spécifié par l'utilisateur puis exécutons la dernière commande du script. Dans tous les cas, la commande "More" affiche le résultat tel que nous l'avons voulu du contenu de notre fichier.

- Vérifier la présence d'une chaîne de caractères :

Dans un nouveau fichier Batch copiez ce contenu :

```
@echo off
@echo off
findstr "212.37.221.109 Microapp.com" %systemroot%\system32\drivers\etc\hosts
if %errorlevel%==0 goto Msg
echo 212.37.221.109 Microapp.com >> %systemroot%\system32\drivers\etc\hosts
goto :eof
:Msg
echo Le fichier Hosts est déjà à jour !
@echo on
```

Le fichier Hosts consigne un certain nombre de sites en faisant le lien entre l'adresse IP et le nom du site. Vous pouvez de cette façon soit faciliter l'accès à un site (il ne sera pas effectué de requête auprès des serveurs DNS) ou en interdire l'accès (en spécifiant à la suite du nom de domaine l'adresse locale de l'ordinateur). Notre script se contente de vérifier l'existence de l'enregistrement spécifié. Si c'est le cas (if %errorlevel%==0) alors nous exécuterons les commandes indiquées sous l'étiquette :Msg. Sinon nous ajoutons la ligne d'enregistrement dans le fichier Hosts et terminons le script par la commande goto:eof.

- Supprimer ou remplacer une chaîne de caractères dans un texte :

Dans un nouveau fichier Batch saisissez ceci :

```
@echo off
for /f "delims=" %%a in ('type "%1"') do call :commande "%%a"
goto :eof
:commande
set ligne=%1
set ligne=%ligne:"=%
@echo %ligne% >>fichier2.txt
@echo on
```

Dans ce script nous nous proposons de supprimer tous les guillemets... Saisissez simplement le nom du script suivi du nom du fichier à modifier. Une version expurgée sera automatiquement générée. Nous nous servons de la commande "Type" pour afficher le contenu du fichier. Pour chaque ligne affichée par la commande "Type" nous créons une variable nommée %ligne%. Dans cette variable nous opérons une substitution en remplaçant tous les signes " par rien ("=). La nouvelle ligne une fois formatée, nous l'"empilons" dans un fichier nommé Fichier2.txt et ce en nous servant de la commande "Echo".

- Retrouver une chaîne de caractères dans différents fichiers :

Dans un nouveau fichier Batch copiez ce contenu :

```
@echo off
set dossier=%1
set filtre=%2
set chaîne=%3
for /f "tokens=*" %%a in ('dir "%dossier%\%filtre%" /s /b') do (
for /f "tokens=*" %%z in ('type "%%a"^|findstr /I /i /c:"%chaîne%"') do (
@echo %%ass:%%z
)
)
@echo on
```

La syntaxe vous permettant de lancer votre fichier Batch sera celle-ci : Nom_Batch Emplacement Nom_Fichiers Chaîne. Par exemple et en admettant que votre fichier Batch se nomme Test.bat, si vous souhaitez savoir quels sont les fichiers texte placés dans C:\Source qui contiennent la chaîne de caractères "Jean", saisissez :

Test c:\source *.txt Jean

Nous définissons trois variables qui correspondent aux paramètres saisis par l'utilisateur. Un jeton est alloué à chaque ligne trouvée dans la sortie-écran de la commande "Dir". Nous obtenons à chaque fois une variable %f qui sera le nom du fichier. Cette variable est reprise de telle sorte que "Type" puisse afficher le fichier spécifié. Un autre jeton est alloué à chaque ligne trouvée dans le fichier analysé. La commande "Findstr" permet de filtrer que les lignes contenant la chaîne de caractères "Jean". Les deux variables ainsi obtenues %f et %z seront affichés par la commande "Echo".

- Trouver une même occurrence dans plusieurs fichiers :

Ce fichier Batch sait le faire :

```
@echo off
for /f "delims=" %%a in ('findstr /m Microapp.com c:\*.*') do echo "%%a"
@echo on
```

- Une autre manière de supprimer les guillemets ?

Si dans un fichier .txt vous avez ce type de contenu :

ligne, 1, mot, 1
ligne, 2, mot, 2

Ces deux commandes vont nous permettre d'afficher que les chiffres ou que les chaînes de caractères :

for /f "tokens=2,4 delims=," %a in (fichier.txt) do (echo %a %b)
for /f "tokens=1,3 delims=," %a in (fichier.txt) do (echo %a %b)

Si, par contre le contenu du fichier ressemble à celui-ci :

ligne "1" mot "1"
ligne "2" mot "2"

Vous devez vous inspirer de ce fichier Batch :

```
@echo off
setlocal EnableDelayedExpansion
for /f "tokens=*" %%a in ('type "Fichier.txt"') do (
set ligne=%%a
echo !ligne:"=!
)>> temp.txt
for /f "tokens=1,3 delims= " %%a in (temp.txt) do (echo %%a %%b)
del temp.txt
@echo on
```

Tout d'abord nous précisons que nous allons nous servir des variables dynamiques. La commande "Type" permet d'afficher le contenu du fichier. À chaque ligne trouvée est allouée une variable %a. Nous redirigeons chacune de ces lignes dans un fichier nommé Temp.txt. Nous nous servons de la commande "Echo" et de la variable %ligne%, mais placée, cette fois-ci, entre deux points d'exclamation. Au passage, nous opérons une substitution en remplaçant tous les guillemets trouvés par rien (echo !ligne:"=!). Il suffit après de reprendre le fichier Temp.txt afin d'afficher son contenu "transformé". Par mesure de "propreté" nous supprimons le fichier temporaire dont nous nous sommes servis.

- Afficher les informations d'un fichier :

Dans un nouveau fichier nommé Script.bat copiez ce contenu :

```
@echo off
set nom=%1%
for /f %%a in ('dir "%nom%" /s /b /-c /a:-d-s-h') do echo %%~za octets
for /f "skip=4 tokens=1,2" %%a in ('dir "%nom%" /tc /-c /a:-d-s-h^|find /v "(s)"') do echo Créé le %%a %%b
for /f "skip=4 tokens=1,2" %%a in ('dir "%nom%" /tw /-c /a:-d-s-h^|find /v "(s)"') do echo Modifié le %%a %%b
for /f "skip=4 tokens=1,2" %%a in ('dir "%nom%" /ta /-c /a:-d-s-h^|find /v "(s)"') do echo Dernier accès le %%a
%%b
@echo on
```

- Saisissez le nom du Batch suivi du nom du fichier à analyser :

Pour bien comprendre comment fonctionne ce script, comparez ces deux commandes :

```
@echo off
dir "fichier.txt" /tc /-c /a:-d-s-h
@echo on
```

```
@echo off
dir "fichier.txt" /tc /-c /a:-d-s-h |find /v "(s)"
@echo on
```

La seconde commande a biffé toutes les informations contenant la chaîne de caractères "(s)" comme celle-ci : "1 fichier(s)". Reste maintenant à supprimer les lignes d'informations d'en-tête. C'est le pourquoi de l'emploi du mot-clé "skip=4". De cette façon, nous n'analysons que la seconde ligne de la sortie-écran produite par la commande "Dir". Je précise que le chiffre de 4 lignes ne correspond pas à la réalité, mais, curieusement, cela fonctionne aussi bien que si nous demandons de "sauter" les cinq premières lignes ?

- Modifier un texte :

Imaginons un fichier nommé Test.txt ayant ce contenu :

Colonne Colonne Colonne
Colonne Colonne Colonne
etc.

Nous souhaitons placer devant chaque colonne un numéro. Dans un nouveau fichier Batch, copiez ceci :

```
for /f "tokens=1,2,*" %%a in ('type test.txt') do echo 1 %%a 2 %%b 3 %%c > test.txt
```

Nous balayons à l'aide de la commande "For" le fichier spécifié en envoyant à chaque bloc rencontré le numéro de la colonne. Si vous souhaitez revenir en arrière, créez un autre fichier Batch avec ce contenu :

```
for /f "tokens=2,4,6" %%a in ('type test.txt') do echo %%a %%b %%c > test.txt
```

Mettre une ligne sur une colonne. Inspirez-vous de ce modèle de script :

```
@echo off
for /f "tokens=1,2,*" %%a in ('type test.txt') do (
echo %%a
echo %%b
echo %%c
) >> sortie.txt
@echo on
```

Dans ce cas, le principe consiste à récupérer chaque bloc de donnés, puis de se servir de la commande "Echo" pour créer un autre fichier nommé Sortie.txt. L'utilisation des parenthèses permet d'entasser les lignes dans un même fichier.

- Lire le contenu d'un fichier texte :

Il y a plusieurs méthodes possibles. Dans un nouveau fichier Batch, copiez ceci :

```
@echo off
for /f "delims=" %%a in (test.txt) do echo %%a
for /f "delims=" %%a in ('type test.txt') do echo %%a
for /f "delims=" %%a in ('more ^< test.txt') do echo %%a
@echo on
```

Comme nous le voyons, le contenu est identique, mais certaines commandes s'avèrent plus efficaces dans le cas de scripts complexes.

- Supprimer toutes les lignes vides d'un fichier :

Vous pouvez vous inspirer de ce type de fichier Batch :

```
@echo off
for /f "delims=" %%a in ('type fichier.txt') do echo.%%a>>fichier1.txt
@echo on
```

Une nouvelle version du fichier sera créée...

- Afficher la première ou la dernière ligne d'un fichier :

Dans un nouveau fichier Batch, copiez ceci :

```
@echo off
set "ligne1="
for /f "delims=" %%a in ('type fichier.txt') do (
if not defined ligne1 set ligne1=%%a
)
echo %ligne1%
@echo on
```

Nous définissons une variable nommée" %ligne1% que nous ne définissons pas. Pour chaque ligne rencontrée par la commande "For" nous définissons la variable %ligne1% sur la première ligne puis reprenons la commande pour la seconde ligne présente dans le fichier. Mais puisque notre variable est déjà définie, nous ne lui attribuerons plus d'autre valeur. Et donc nous afficherons que le contenu de la première ligne. En sens inverse, vous pouvez utiliser ce type d'astuce :

```
@echo off
for /f "delims=" %%a in ('type fichier.txt') do set "dernière=%%a"
echo %dernière%
@echo on
```

Là, c'est beaucoup plus simple ! Nous nous contentons d'afficher la dernière valeur trouvée pour la variable %dernière% et c'est forcément la dernière ligne parcourue par la commande "For".

- Afficher un certain nombre de lignes :

Dans un nouveau fichier Batch, copiez ceci :

```
@echo off
set "lignes=%1"
set B=-1
set "C="
for /f "delims=" %%a in ('type fichier.txt') do (
set/a B+=1 & for /f %%z in ('echo %%B%%') do (
if "%%z"=="%lignes%" set C=1
)
if not defined C echo %%a
)
@echo on
```

Il suffit de saisir le nom du script suivi du nombre de premières lignes que nous souhaitons afficher. Nous posons trois variables : %lignes%, %B% et %C%. Cette dernière n'étant pas définie. La variable %lignes% correspond au nombre de lignes saisies par l'utilisateur à la suite du nom du Batch. La variable %B% reçoit une valeur négative au cas où l'utilisateur ne veuille afficher qu'une ligne. Un balayage du fichier est effectué par la commande "For" et nous passons à l'opération suivante : Nous incrémentons la variable %B% d'un pas de un et nous affichons la valeur obtenue. Un second balayage est alors effectué : Si la variable obtenue correspond au nombre de lignes spécifié par l'utilisateur alors nous définissons la variable %C% sur 1. À l'inverse, si la variable %C% n'est pas définie alors nous affichons la ligne en cours. En d'autres termes, si %z n'est pas encore égale à %lignes% alors nous affichons la ligne en cours. Quand la variable %z sera égale à %lignes% alors nous n'afficherons pas la ligne en cours. Étant donné que nous ne visualiserons que les lignes affichées par la commande "Echo" nous garderons l'impression que seules ces lignes ont été parcourues. En sens inverse, voici une suggestion :

```
@echo off
set "lignes=%1"
for /f %%a in ('find /c /v "" ^< fichier.txt') do set/a Suppression=%%a-lignes
for /f "delims=" %%a in ('more /e +%Suppression% ^< fichier.txt') do echo %%a
@echo on
```

Dans ce cas nous affichons les deux dernières lignes. Dans un premier temps nous définissons une variable nommée %suppression%. La commande Find /c /v "" nous permet d'obtenir le nombre de lignes qui ne sont pas vides. La valeur de %suppression% s'obtient à faisant la soustraction du nombre de lignes par le nombre de lignes à afficher. La commande "More" nous permet de n'afficher que la première ligne à partir de la valeur de %suppression%. Si, par exemple, notre fichier compte 11 lignes et que nous souhaitez afficher que les deux dernières lignes, le calcul opéré par le Batch est le suivant : 11 - 2 = 9. Nous afficherons donc le contenu du fichier à partir de la neuvième ligne et donc deux lignes seront "relayées" par la commande "Echo".

- Afficher une ligne précise :

Dans un nouveau fichier Batch copiez ce contenu :

```
@echo off
set Numéro=%1
set "ligne="
set/a Numéro-=1
for /f "delims=" %%a in ('more/e +%%Numéro%% ^< fichier.txt') do (
if not defined ligne set "ligne=%%a"
)
echo %ligne%
@echo on
```

Saisissez le nom du script suivi du numéro de ligne à afficher. Là aussi, saisissez cette commande : echo /e +2 fichier.txt. Nous afficherons le contenu de Fichier.txt à partir de la seconde ligne. Nous effectuons un balayage à l'aide de la commande "For". À chaque fois nous essayons de définir la variable %ligne% sur la ligne en cours. Mais comme la variable %ligne% est définie dès la première ligne trouvée nous n'affichons que cette première ligne. Et comme la première ligne trouvée correspond bien à celle à partir de laquelle nous exécutons la commande, c'est la ligne correcte qui sera relayée par la commande "Echo".

- Paramétrer les sorties :

Comparez la mise en page produite par ces trois fichiers Batch :

```
echo BLABLA >sortie1.txt
echo BLABLA >>sortie1.txt
echo.|set /p=BLABLA >sortie2.txt
echo.BLABLA >>sortie2.txt
echo BLABLA ^BLABLA >sortie3.txt
```

Le signe ^ provoque un retour chariot.

- Renvoyez le code d'erreur d'un fichier Batch :

Dans un nouveau fichier Batch copiez ce contenu :

```
@echo off
dir *.doc
for %%a in (5 4 3 2 1 0) do if errorlevel==%%a (
set code=%%a
goto erreur
)
:erreur
echo Le code renvoyé par la commande est %code%
@echo on
```

- Retrouver la version d'Internet Explorer :

Nous avons adapté un script proposé par Jerold Schulman à partir de son site Web : www.jsiinc.com. Dans un nouveau fichier Batch copiez ce contenu :

```
@echo off
set valeur=reg query "HKEY_LOCAL_MACHINE\Software\Microsoft\Internet Explorer" /v Version
set version=findstr /I /L /C:"REG_SZ"
for /f "tokens=1,2,3" %%a in ('%valeur%^|%version%') do @echo La version d'IE est : %%c
@echo on
```

Cette information est consignée dans le Registre Windows. Nous interrogeons donc le Registre en parcourant cette arborescence : HKEY_LOCAL_MACHINE\Software\Microsoft\Internet Explorer. Nous définissons une variable nommée %valeur% qui sera la sortie-écran de la commande "Reg query". Nous posons ensuite une variable nommée %version% qui sera la commande "Findstr" paramétrée pour rechercher la chaîne de caractères "REG_SZ". Enfin, nous effectuons un

balayage dans la sortie-écran affichée par la commande "Findstr" (remplacée par la variable %valeur%) et nous ne relayons que le troisième bloc de données trouvé.

- Un script pour redémarrer votre ordinateur :

Dans un nouveau fichier Batch, copiez ce contenu :

```
@echo off
set /p question=Redémarrez votre ordinateur [O/N]?
if /i not "%question%"=="O"
@shutdown -r
@echo on
```

Ce sera l'utilisateur qui définira la valeur de la variable %question%. Si cette dernière est égale à o alors nous lançons la commande "Shutdown". Sinon, nous terminons le script en utilisant la commande "Goto :eof".

- Calculer le temps passé sur une application :

C'est une solution simple qui est proposée ici, mais il est très facile d'améliorer ce script... Dans un nouveau fichier Batch copiez ce contenu :

```
@echo off
echo Lancement de Notepad >> journal.txt
for /f "tokens=*" %%a in ('date /t') do echo la date est le %%a >> journal.txt
for /f %%a in ('time /t') do echo l'heure est le %%a >> journal.txt
notepad
echo "Appuyez sur une touche afin d'enregistrer la date et l'heure de fermeture..." & pause >nul
echo Fermeture de Notepad >> journal.txt
for /f "tokens=*" %%a in ('date /t') do echo la date est le %%a >> journal.txt
for /f %%a in ('time /t') do echo l'heure est le %%a >> journal.txt
@echo on
```

À chaque fois que nous lançons le Batch nous ouvrirons le Bloc-notes Windows. Tant que nous travaillerons sur cette application, le fichier batch restera ouvert. La fermeture du programme oblige le script à afficher un message vous demandant d'appuyer sur une touche afin de continuer son exécution. Ces évènements seront consignés dans un fichier nommé Journal.txt. Ainsi, à tout moment il vous sera possible de consulter le temps passé sur tel ou tel document.

Les fichiers Batch

Un fichier Batch est un fichier qui regroupe une suite logique de commandes MS-DOS, et possédant l'extension ".bat" et est directement exécutable par le système. Les commandes sont enregistrées ligne par ligne et seront exécutées séquentiellement. La programmation Batch nécessite une connaissance minimum de l'environnement Dos. En fait, un fichier Batch contient simplement une suite de commandes que vous pourriez taper sous l'invité (prompt) du Dos, chaque nouvelle ligne du fichier correspondant à une nouvelle commande. Néanmoins, certaines commandes ne sont utilisables que dans les fichiers Batch du fait de leur inutilité dans l'environnement de commande Dos. C'est ce qui permet d'automatiser certaines tâches. Leur utilité est, par exemple, quand il faut répéter toujours la même série de commandes. À titre d'exemple, nous pourrions évoquer le changement de répertoire et peut-être aussi la commande Format qu'on fait souvent suivre de la commande CHKDSK pour vérifier si la disquette a bien été formatée.

Table des matières :

À propos de l'auteur...

Stéphane Grare, Concepteur et développeur C#, Stéphane Grare, passionné par la programmation informatique, est l'auteur de plusieurs tutoriels et livres blancs sur différents langages de programmation informatiques. Il est aussi à l'initiative du projet Simply, une gamme d'applications dédiées à la bureautique.

Niveau : Intermédiaire
Catégorie : Programmation
Configuration : Windows

